Planning Simultaneous Perception and Manipulation

by

Claudio Zito

A thesis submitted to
the University of Birmingham
for the degree of
Doctor of Philosophy

Supervisors: Jeremy Wyatt, Rustam Stolkin

May 25, 2016

Abstract

This thesis is concerned with deriving planning algorithms for robot manipulators. Manipulation has two effects, the robot has a physical effect on the object, and it also acquires information about the object. This thesis presents algorithms that treat both problems.

First, I present an extension of the well-known piano mover's problem where a robot pushing an object must plan its movements as well as those of the object. This requires simultaneous planning in the joint space of the robot and the configuration space of the object, in contrast to the original problem which only requires planning in the latter space. The effects of a robot action on the object configuration are determined by the non-invertible rigid body mechanics. To solve this a two-level planner is presented that coordinates planning in each space.

Second, I consider planning under uncertainty and in particular planning for information effects. I consider the case where a robot has to reach and grasp an object under pose uncertainty caused by shape incompleteness. The main novel outcome is to enable tactile information gain planning for a dexterous, high-degree of freedom manipulator with non-Gaussian pose uncertainty. The method is demonstrated in trials with both simulated and real robots.

Dedication

To my family.

Well, what've you got?

Egg and Spam;

Egg, bacon and Spam;

Egg, bacon, sausage and Spam;

Spam, bacon, sausage and Spam;

Spam, egg, Spam, Spam, bacon and Spam;

Spam, Spam, Spam, egg and Spam;

Spam, Spam, Spam, Spam, Spam,

Spam, baked beans, Spam, Spam, Spam and Spam;

Lobster Thermidor aux crevettes with a Mornay sauce,

garnished with truffle pâté, brandy and a fried egg on top, and Spam.

(Monty Python's Flying Circus. SPAM sketch. 15 Dec 1970)

Declaration

I hereby declare that I composed this thesis entirely myself and that it describes my own research.

I also declare that, to my knowledge, this is the only thesis ever written in which the acronymous of the title is the same as the lyrics of a Monty Python's song.

Acknowledgements

The most important acknowledgement must go to my two supervisors: Prof. Jeremy Wyatt and Dr. Rustam Stolkin for providing encouragement and support even in the most arduous stages of my PhD. For reading the draft(s) of my thesis and for the helpful feedback. But I would like to single out Jeremy for providing a decent salary along these last five years and a countless meals he has bought while I was locked in the lab testing.

A special thank it is due to Marek Kopicki for his friendship along with the load of work resulted more affordable. I am also thankful to all the colleagues for all the events, discussions and good time we shared together. In particular, Loretta Ilaria Mancini who has taken care of me as an "older" sister. A special thank is also due to Prof. Antonio Bicchi who coined the term SPAM for manipulation problems.

I also want to thank my RMSG members Richard Dearden, Hamid Deghani and Xin Yao who followed the progress of my research, reading all my progress report accurately and provided with helpful comments and suggestions.

My final thank goes to Chiara Busa for the patience she demonstrated in my writing up period and the bottomless support she gave me.

Contents

List of Figures

List of Tables

List of Abbreviations

DoF(s)	Degree(s) of freedom	Sec. 3.2.1
DM(s)	Decision maker(s)	Sec. 4
FM(s)	Forward model(s)	Sec. 2.9.1
IG	Information gathering approach	Chap. 6
ISMDP(s)	Information state Markov decision process(es)	Sec. 4.4.1
MDP(s)	Markov decision process(es)	Sec. 4.3.1
MGA	Maximum grip aperture	Sec. 2.7
MT	Movement time	Sec. 2.7
TPA	Time to peak acceleration	Sec. 2.7
TPD	Time to peak deceleration	Sec. 2.7
TPV	Time to peak velocity	Sec. 2.7
PI	Policy iteration algorithm	Sec. 4.3.4.2
POMDP(s)	Partially observable Markov decision process(es)	Sec. 4.4.1
PRM(s)	Probabilistic roadmap(s)	Sec. 3.3.2.2
RRT(s)	Rapidly-exploring random tree(s)	Sec. 3.3.2.1
VI	Value iteration algorithm	Sec. 4.3.4.1

List of Symbols

$\times$	Euclidian product	Sec. 3.2.3
$\lvert\cdot\rvert$	Cardinality of a set	Sec. 4.3.1
$\varnothing$	Empty set	Sec. 3.3.1
$\mathbf{A}$	Set of discrete actions	Sec. 4.3.1
$\mathcal{A}$	Operational workspace occupied by robot's bodies	Sec. 3.3.1
$\mathbf{B}$	Belief space	Sec. 4.4.2
b_t	Belief state at time step t	Sec. 4.4.2
$\mathbf{C}$	Configuration space for a robot, joint space	Sec. 3.2.3
$\mathbf{C}$-space	Same as $\mathbf{C}$	Sec. 3.3.1
$\mathbf{C}_{free}$	Configuration space free from obstacles	Sec. 3.3.1
$\mathbf{C}_{obs}$	Configuration space occupied by obstacles	Sec. 3.2.4
$\mathbf{C}_{work}$	Workspace for a robot in configuration space	Sec. 3.2.4
$\mathbf{E}$	Set of edges in a graph	Sec. 3.3.2.1
$\mathbf{G}$	Graph	Sec. 3.3.2.1
h_t	History of a decision theoretic system at time step t	Sec. 4.4.2
$O(\cdot)$	Upper bound on the growth rate of a function	Sec. 4.3.4.2
$\mathbf{O}$	Set of observations	Sec. 4.4.1
Π	Set of probabilities	Sec. 4.3.1
π	Action-selection policy	Sec. 4.3.3
Pr	Probability	Sec. 4.3.1
$\mathcal{Q}(s,a)$	State-action value function	Sec. 4.3.4.3
$\hat{\mathcal{Q}}$	Approximated state-action value function	Sec. 4.3.4.3
$\mathcal{Q}(b,a)$	Belief-action value function	Sec. 4.3.4.3
$\mathcal{Q}*$	Belief-action value function following the optimal policy	Sec. 4.4.2
q	Vector of joint angles	Sec. 3.2.3
q_I	Initial configuration of a robot in joint angles	Sec. 3.3.1
q_G	Goal configuration of a robot in joint angles	Sec. 3.3.1

$\mathbb{R}$	Set of real numbers	Sec. 3.2.3
$\mathbb{R}^r$	Euclidian product of linear intervals	Sec. 3.2.3
$\mathcal{R}$	Expected immediate reward function	Sec. 4.3.1
$\mathbf{R}(b_1)$	Reachable belief space from b_1	Sec. 4.5.1
$\mathbf{R}^*(b_1)$	Optimally reachable belief space from b_1	Sec. 4.5.1
$\hat{\mathbf{R}}(b_1)$	Approximated reachable belief space from b_1	Sec. 4.5.1
$\mathbb{S}$	Unit circle	Sec. 3.2.3
$\mathbf{S}$	Set of discrete states	Sec. 4.3.1
$\mathbf{T}$	Set of time steps	Sec. 4.3.1
$\mathbb{T}^p$	Euclidian product of unit circles, p-torus	Sec. 3.2.3
$\mathcal{T}$	State transition function	Sec. 4.3.1
t	Time step	Sec. 4.3.1
τ	Global trajectory or path for a robot from initial to goal configurations	Sec. 3.3.1
τ_{local}	Local trajectory of path for a robot between two vertices in a graph	Sec. 3.3.2.2
Θ_i	Joint variable for joint i^{th}	Sec. 3.2.3
$\mathbf{U}$	Set of continuous actions	Sec. 5.3.1
$\mathbf{V}$	Set of vertices in a graph	Sec. 3.3.2.1
$\hat{\mathbf{V}}$	Neighbouring set of vertices for a particular q	Sec. 3.3.2.2
V	Utility or value function	Sec. 4.3.3
V^π	Utility or value function following policy π	Sec. 4.4.2
V^*	Utility or value function following the optimal policy	Sec. 4.4.2
$\mathbf{W}$	Operational workspace in world's coordinates	Sec. 3.2.5
$\mathbf{W}_{obs}$	Operational workspace occupied by obstacles	Sec. 3.3.1
$\mathbf{X}$	Set of continuous states	Sec. 5.3.1
$\mathcal{Y}$	Observational function	Sec. 4.4.1

Chapter 1

Introduction

This thesis is concerned with deriving algorithms for robot manipulators. First, Sec. 1.1 explains why the grasping and manipulation problem is challenging in robotics. That is for several reasons: (i) it is formulated in a high-dimension, continuous space; (ii) complex relationship between robot's actions and their effects on the object to be manipulated; and (iii) imperfect sensing capabilities of the robot.

Section 1.2 explains the main hypotheses underpinning this thesis, which results on the formulation of a *simultaneous perception and manipulation* (SPAM) problem, and lists the contributions of this thesis. Section 1.3 describes the proposed approaches to solve two problems in manipulation: (i) simultaneously planning movements for a robot pushing an object as well as those of the object (Sec. 1.3.1) and (ii) planning dexterous reach-to-grasp trajectories which maximise information gain under conditions of perception uncertainty with respect to the location of the object to be grasped (Sec. 1.3.2).

Finally, Sec. 1.4 describes the working scenarios used in this thesis to test and evaluate the proposed algorithms. The chapter is concluded by summarising the context of the other chapters that compose this thesis (Sec. 1.5).

1.1 Motivation

This thesis is concerned with extending path planning algorithms to cope with continuous or uncertain domains. In particular, the domain of application is the one of dexterous grasping and manipulation in robotics. The aim is to derive controllers for robot manipulators in embedded systems, in which a robot takes as input the state of the world (perception) and computes as output manipulative actions that themselves affect the state of the world. A visual representation of an embedded system is shown in Fig. 1.1.

Even in ideal conditions, such as *structured environments* where a robot has a complete model of the environment and perfect sensing abilities, the problems of robotic grasping and manipulation are not trivial. In robotic manipulation and grasping, by complete model of the environment we mean that physical and geometric properties of the world, such as pose, shape, friction parameters and the mass of the object we wish to manipulate, are exactly known. However, the manipulation and grasping problem is challenging for several reasons. First, the object to be manipulated is indirectly controlled by contacts with a robot manipulator (e.g. pushing by a contacting finger part), and an *inverse model* (IM), which computes an action to produce the desired motion or set of forces on the object, does not exist. Sometimes *forward models* (FM) may be fully or partially known, even where IMs are not available. In such cases, an FM can be used to estimate the next state of a system, given the current state and a set of executable actions. This enables planning to be achieved by imagining the likely outcomes from all possible manipulative actions, and then choosing the action which achieves the most desirable end state. However the manipulation and grasping problem is typically defined in continuous state and action spaces, hence it is computationally intractable to build an optimal sequence of actions, or plan, by exploring all possible action-state combinations.

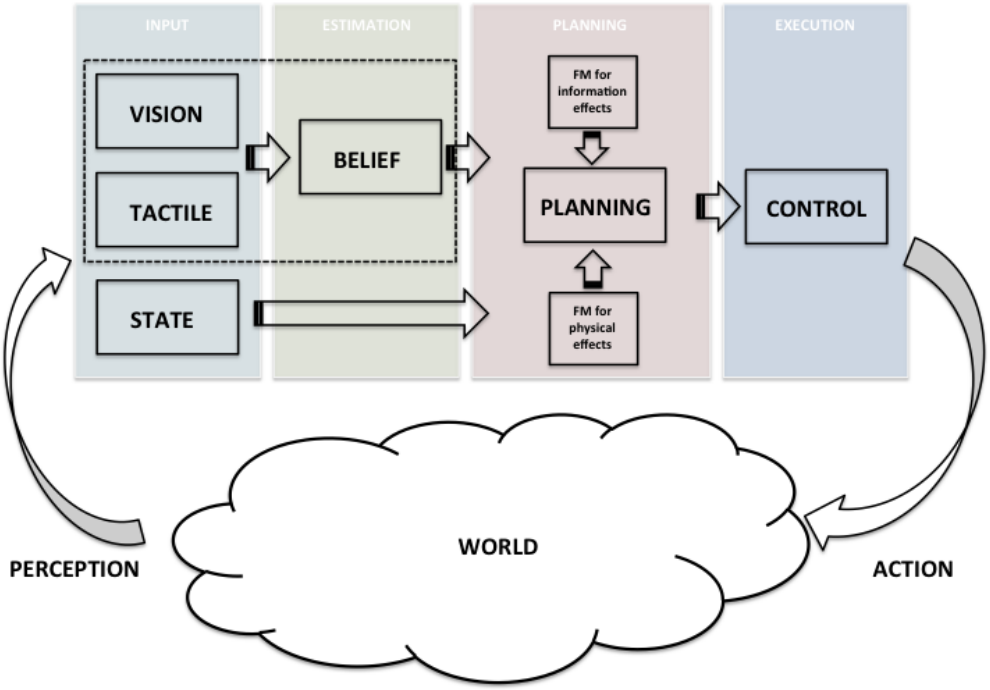


Figure 1.1: The picture represents an embedded system where a robot (or agent) interacts with the environment. The robot takes as input the state of the world, or a noisy observation of it, and computes as output an action which will affect the state of the world. The architecture of the system is composed of 4 components: input, pose estimation, planning and control. If perfect sensing abilities are assumed, the input is a complete description of the state and thus no state estimation is needed. The planning block includes two modules which are the major contribution of this thesis, namely: i) a forward model (FM) for predicting the expected informational gain and ii) a FM for predicting the expected physical effects. The arrows show the flow of information and the interaction of the system with the external world.

Even more challenging is the problem of grasping and manipulation in *unstructured environments*, where these ideal conditions do not exist. There are several reasons why an agent may fail to build a complete description of the state of the environment: sensors are noisy, robots are difficult to calibrate, actions' outcomes are unreliable due to unmodelled variables (e.g. friction, mass distribution). Uncertainty can be modelled in several ways, but in case of manipulation there are typically two types of uncertainty:

- *Uncertainty in physical effects*: occurs when the robot acts on external bodies via

physical actions (e.g., contact operations). This interaction transforms the current state of the world according to physical laws which are not fully predictable. For example, a pushed object may slide, rotate or topple with complex motions which are extremely difficult to predict, and involve physical parameters which may not be known. We can think of this as an uncertainty on future states.

- *Uncertainty in sensory information*: occurs when some of the quantities that define the current state of the world are not directly accessible to the robot. Thus the necessity to develop strategies to allow the robot to complete tasks in partial ignorance by recovering knowledge of its environment. In such cases there is uncertainty about how much new information will be yielded during the execution of a new robotic action.

Although the ongoing development of more accurate sensors and robots may increase robustness, uncertainty of various kinds will always remain. The approaches presented in this thesis aim at coping with, rather than eliminating, such uncertainty. There is also evidence that the *sensorimotor control* in humans models actions as decisions, in order to maximise some reward function associated with the motor outcome [Körding & Wolpert, 2004]. Associated models of optimal decision making provide explanations of how humans compensate for uncertainty. So far there have only been limited investigations of decision making for complex manipulation actions, however experimental results confirm that humans cope with uncertainty in simple manipulations such as two-finger grasping by adopting grasp strategies that increase the chance of a stable grasp at the first contact [Christopoulos & Schrater, 2009].

There are two classes of planning approaches that are related to the work in this thesis:

- *Path planning* provides methods for planning trajectories for robot manipulators in deterministic, but continuous domains [see LaValle, 1998]. Path planning was

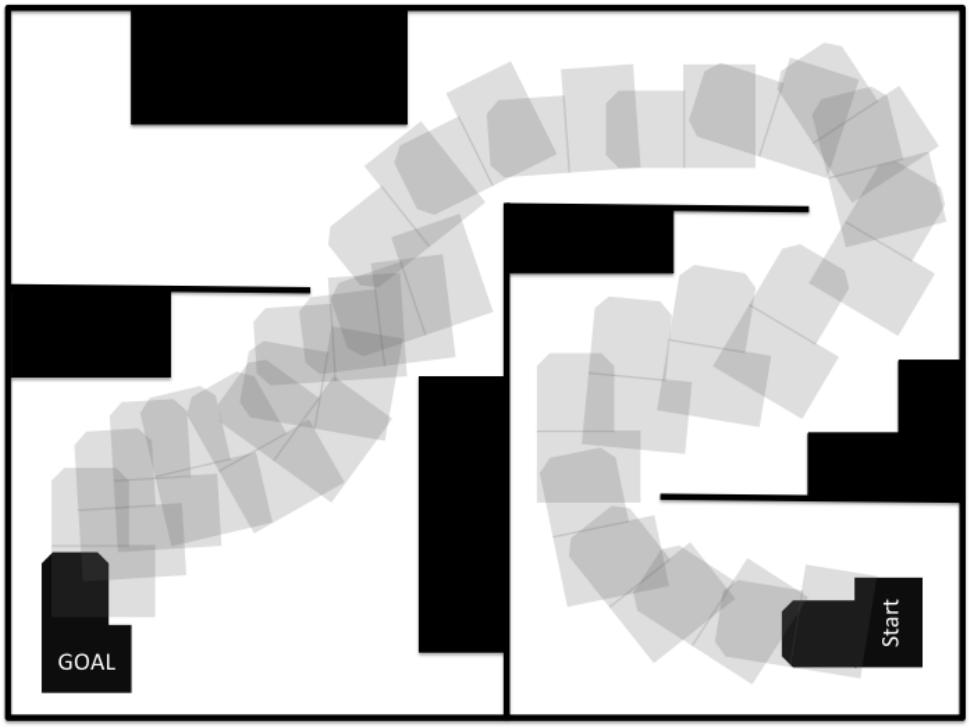


Figure 1.2: Graphical representation of the piano mover's problem. Path planning was originally concerned with problems such as how to move a piano from one room of a house to another without colliding with obstacles or bounding walls.

originally concerned with problems such as how to move a piano from one room of a house to another without colliding with obstacles or bounding walls (Fig 1.2). In manipulation and grasping however, the object to be moved is indirectly controlled by contact with a robot manipulator. This touches on the general problem of how to plan actions in one space (e.g., the joint space of an arm) which affects on another space (e.g. the configuration space of an object) where there is no inverse model of the effects. In addition, if the robot does not have perfect perception of the current state, it may fail to achieve the planned contacts and therefore jeopardise the action's outcome;

- *Planning under uncertainty* provides methods for formalising decision problems in stochastic, but discrete domains, in which an agent takes as input the state

of the world and generates as output actions which themselves affect the state of the world. Such frameworks can be naturally extended to deal with uncertainty in partially observable environments, e.g. Partially Observable Markov Decision Processes (POMDPs). The POMDP framework embraces a large range of practical problems, but solving a POMDP is intractable [Papadimitriou & Tsitsiklis, 1987].

This thesis investigates hybrid models for planning in continuous and non-deterministic domains.

1.2 Hypotheses and contributions

The main idea underpinning this thesis will be that:

> *in order to deal with imperfect information, we should derive algorithms which can model and explicitly reason about uncertainty. Manipulation has two effects. A robot action has an effect on the object configuration determined by model-free approach and, in uncertain domains, a robot action can also acquire information about the object. So we should enable our robot to plan manipulative actions by accounting for physical and informational effects, and recover from incorrect assumptions.*

This combination of (active) perception and manipulation tasks makes this a problem of SIMULTANEOUS PERCEPTION AND MANIPULATION (SPAM). In general manipulation problems are problems of sequential decision making, in continuous or uncertain domains, and could be posed either as learning or planning problems. This thesis explores how such problems can be posed as planning problems. In addition this thesis presents a planning system for solving this formulation of the SPAM problem (SPAM-PLAN), which combines the benefits of both motion planning algorithms and decision-theoretic frameworks, in order to extend planning algorithms for robot manipulators to cases

where physical and informational effects must be modelled.

In particular, two different instances of the SPAM problem are investigated:

- *Planning for physical effects*: which considers the case of how to plan a sequence of push operations, for a robot manipulator equipped with a single finger, to cause a desired physical effect on the environment, i.e. to move an object from an initial pose to a desired one. This problem is presented in a 6D domain where the object to be pushed is free to slide, flip and rotate. No inverse model is known, and the state space is continuous (i.e. all the possible configurations of the object), as well as the action space (i.e all the possible push trajectories for the robot manipulator). In this problem it is assumed that the robot has complete observability of the state space to determine the exact pose of the object.
- *Planning dexterous grasps for informational effects*: which considers the case of how to plan reach to grasp trajectories, for a dexterous manipulator equipped with an anthropomorphic robot hand, which are designed to increase information about the grasped object, in addition to achieving a grasp. In particular, this work copes with object pose uncertainty, and it has been extended also to include shape incompleteness of the object to be grasped. This thesis presents a set of algorithms that use a combination of information gain planning, hierarchical probabilistic roadmap (PRM) planning, and belief updating from tactile information for objects with non-Gaussian pose uncertainty in 6D domains.

Several contributions are made, which include:

- The SPAM problem is naturally defined as decision making. This thesis explores the computational limitations of such a framework and proposes an alternative formulation (Chap. 4).
- Investigate hybrid models to combine the benefits of both path planning algorithms

and decision making frameworks. This is done by extending path planning algorithms for dexterous robot manipulators in continuous, high-dimensional spaces under state and physical effects uncertainty in 6D domains (Chap. 4).

- Show how to extend existing motion planning algorithms to build plans for push operations in continuous, but non-deterministic 6D domains (Chap. 5). The key intuition is to take advantage of the exploring capabilities of motion planner algorithms (here an RRT) to span the configuration space of the object we wish to manipulate (Sec. 5.3.1). This leads to a collision-free trajectory that the object should follow in order to be placed in a goal region. The problem of finding the appropriate sequence of pushes to generate the desired motion of the object is a decision making problem in stochastic, but continuous domains.

- Show how to efficiently solve the associated decision making problem for the push planner in continuous state and action spaces (Sec. 5.3.2). This is done by treating each RRT node extension as an optimisation problem under the assumption of deterministic state transitions (here I used a physics-based engine). The optimisation problem is challenging because there is no inverse model available and an infinite number of possible push trajectories. The optimiser solves the problem ignoring the uncertainty in physical effects, however if the expected outcome differs from the planned one, a re-planning stage is executed. It is assumed that the robot has direct access to the state of the environment, so that it can track the motion of the object while executing a push operation.

- Present an efficient sample-based method to span the action space on-the-fly for the associated optimisation problem for planning push operations (Sec. 5.3.2). This allows us to build a finite set of push trajectories given a particular pose of the object to be pushed. This set of actions is proved to generate a minimum number of pushes that allows us to move the object in any direction.

- Show the effectiveness of the algorithm in a simulated environment in which a robot has direct access to the state of the environment, i.e. the configuration of the object, (Sec. 5.4).

- Present a set of sequential re-planning algorithms to plan dexterous reach-to-grasp trajectories in continuous, high-dimensional spaces with non-Gaussian object pose uncertainty in 6D. This is done by using a hierarchical sample-based path planner, here a Probabilistic Roadmap (PRM) planner (Chap. 6).

- Show how to extend such sample-based planners for information effects (Sec. 6.4). Rather than planning in belief space, this work embeds the value of information in the underlying physical space. This is done by assuming that the most likely observation (here tactile contacts or lack of them) will always occurs. Information gain value is computed as the difference in the expected observations between the hypothesised position and each alternative. This results in a sequence of actions that are most likely to generate observations that distinguish a hypothesised state from competing hypotheses while also reaching a goal position.

- Show how to encode non-Gaussian object pose uncertainty in 6D and shape incompleteness in a particle-based belief state (Sections 6.3.2 and 7.3.1).

- Show how to refine the expected object pose by using an observation model for contact sensing by a multi-finger hand that palpates a 3D object to be grasped (Sec. 6.4.1).

- Interpret the noisy contact sensors of the robot hand to efficiently stop a reach-to-grasp trajectory if a contact occurs (Sec. 7.3.4).

- Show how to efficiently plan collision-free dexterous reach to grasp trajectories for non-convex objects described as point clouds. This is done by implementing an efficient KD-tree based collision detection module for point clouds (Sec. 7.3.5).

- Show that this approach enables planning for robot manipulators with 21 DoF and non-Gaussian object pose uncertainty in 6D (Sec. 6.5) and shape incompleteness (Sec. 7.4).

1.3 Proposed approaches

This section presents the core ideas of the algorithms presented in this thesis.

1.3.1 Planning push operations for physical effects

The main aim is to enable a robot manipulator equipped with a single finger to move via push operations an object from an initial pose to a desired one while avoiding collisions with obstacles. The proposed solution is to break the problem into two levels: i) planning motions of the object using a standard RRT planner (see Sec. 3.3.2.1), and ii) planning the actions of a robotic finger to achieve those object motions, using a finger push planner that employs a randomised depth-first search procedure.

At each step the RRT generates a new *candidate node*. The finger push planner then randomly generates N pushes, and one of these is selected which moves the object in a direction which maximises the rate at which the distance to this *candidate node* is reduced. That push is executed in the finger push planner until the distance to the candidate node starts to rise again, at which point N new pushes are generated, and one is selected. This process is repeated until the finger push planner has produced a sequence of finger motions that bring the object within some threshold distance of the candidate node. A new node is then added to the RRT for the object pose at the termination point of the final push. The whole process then repeats with the RRT planner generating a new candidate node. The entire procedure terminates when a sequence of finger pushes have been found that carry the object to within a specified

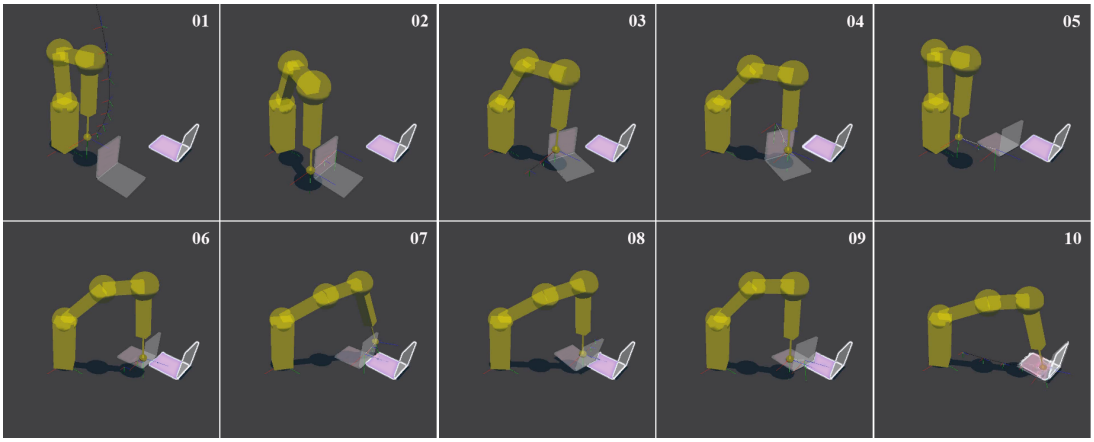


Figure 1.3: The image sequence shows a series of pushes computed by the algorithm in which 8 different push trajectories were randomly selected at each iteration. Image 01 shows the initial configuration of the experiment from which the 5-axis robot executes the solution path. The wire-framed L-shaped object (or polyflap) to the right of the image is a 'phantom' to indicate the desired goal state. With respect to the initial configuration, the goal pose is translated by 28 cm and rotated by 90 degrees. The manipulator first executes a sideways push, as shown in images 02 and 03, and then pushes against the vertical face of the polyflap to make it tip over (images 04-05). In images 06 and 07 the robot executes additional sideways pushes to correct the orientation. In images 08-10, the robot pushes the polyflap on its vertical face to move it onto the desired goal pose.

threshold distance of its target pose.

The resulting algorithm has some useful distinguishing properties. It uses only a forward model (a physics engine) to compute a push plan. Furthermore, it is robust against becoming trapped in local minima. Because of the complex relationship between these poses and the pushes that give rise to them, it is very easy for a conventional RRT approach to get stuck in a local minimum of poses near to the current pose of the target object. Interleaving the additional technique of the randomised depth-first search for pushes, avoids these local minima, and enables the RRT to continue growing new nodes towards the goal position using substantial step sizes. In addition, uncertainty associated with prediction can be overcome by re-planning if the observed object pose at a node deviates from the planned pose for that node by more than some predefined threshold.

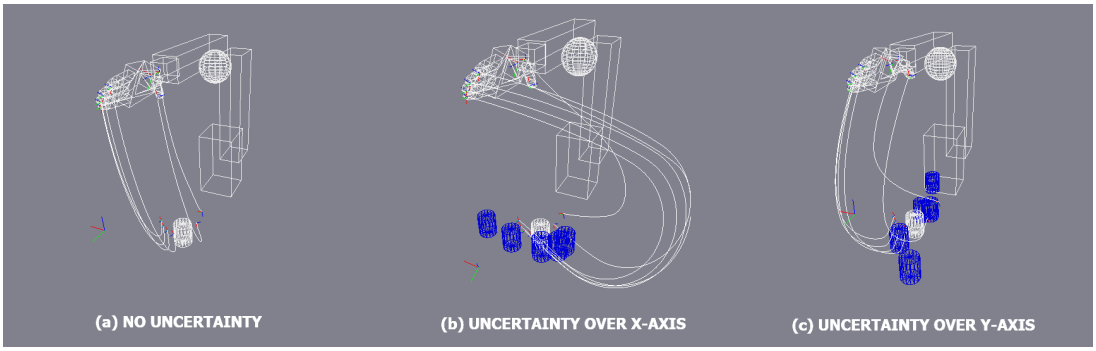


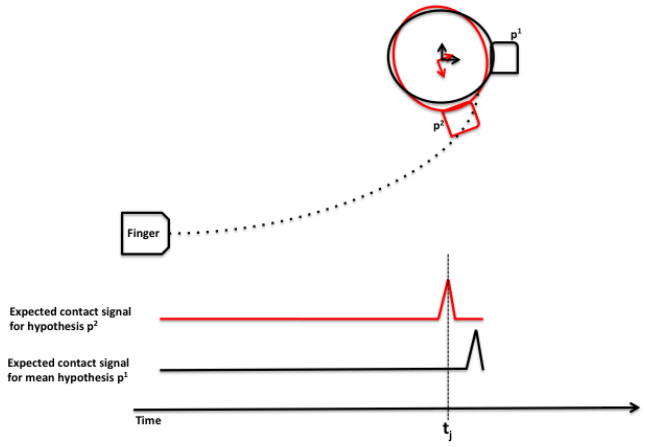
Figure 1.4: Illustration of how the information gain planning strategy behaves with different kind of uncertainty: (a) the system has no uncertainty over the object pose and plans a smooth trajectory directly towards the goal state, which is a pre-shaped side grasp; (b) uncertainty predominantly along the x-axis (red axis); (c) uncertainty predominantly along the y-axis (green axis). Blue cylinders illustrate the sampled object poses whilst the grey cylinder identifies the most likely hypothesis. The example shows significant differences in the approaching trajectory which maximise our expectation of gaining tactile information according to the current belief state. The uncertain regions have as mean pose the hypothesis (grey cylinder) and variance of 4 cm along the major axis of noise and 0.25 cm along the minor axis. No uncertainty along the z-axis (blue axis).

Figure 1.3 shows a series of pushes computed by the algorithm in which $N = 8$ different push trajectories were randomly selected at each iteration. This work has been published in IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems 2012 [Zito et al., 2012b].

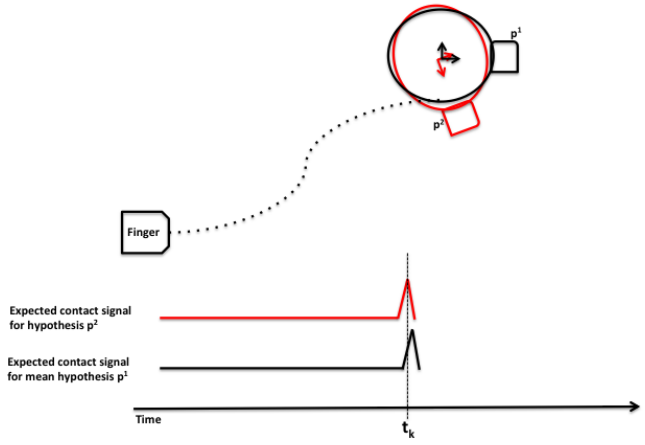
1.3.2 Planning dexterous grasping for information effects

In robot grasping, there is typically uncertainty associated with the location of the object to be grasped. However, if the object is not in its expected location, then a robot equipped with tactile sensors, or torque sensors at finger joints, may gain information to help refine localisation knowledge from tactile contacts (or lack of such contacts) during the execution of a reach to grasp trajectory.

The key intuition of this work is to embed expected (tactile) information value in the underlying physical space; this results in wrapping distances to a non-Euclidian space in



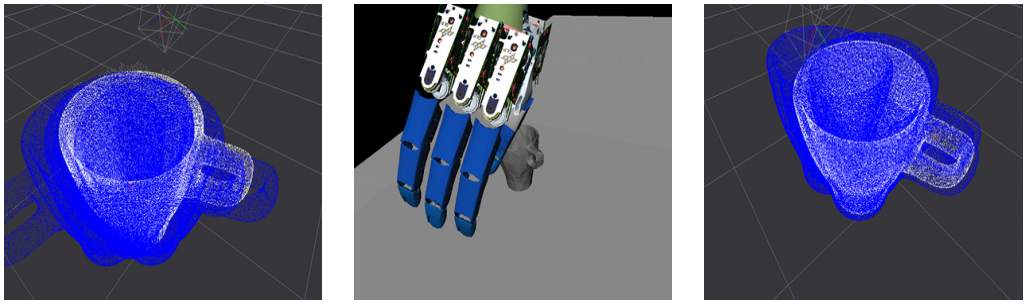
(a)



(b)

Figure 1.5: The figures show the observational model for tactile information. The poses p^1 and p^2 represent two hypothesised configurations of a mug to be grasped. The dotted lines show two possible trajectories for the finger to reach and touch the mug. Hypothesis p^1 represents the expected mean pose for the mug. Figure 1.5(a) shows the expected contact signal for both hypotheses along the trajectory. At time t_j the planner expects to observe a contact if the object is in pose p^2 and no contact for pose p^1. In picture 1.5(b), the planner expects similar observations in both cases at time t_k. Thus the trajectory in 1.5(a) is more likely to distinguish hypothesis p^1 versus p^2 than the trajectory in 1.5(b).

which trajectories that maximise information gain are considered less costly. Figure 1.4 shows a graphical example of information gathering trajectories for three different conditions: (a) the system has no uncertainty over the object pose and plans a smooth, straight trajectory towards the goal state, which is a pre-shaped side grasp; (b) uncertainty prevalently along the x-axis (red axis); (c) uncertainty prevalently along the y-axis (green axis).



(a) **(b)** **(c)**

Figure 1.6: All belief states are the low dimensional belief states sub-sampled from the corresponding high dimensional belief states. Image (a) shows a belief state before a contact occurs, image (b) shows the contact between the thumb of the robotic hand and the object to be grasped. Image (c) shows the *a posteriori* belief state after the contact.

The object pose uncertainty is modelled as a particle filter, in which each particle is a possible hypothesis of the current pose of the object to be grasped. Similarly to [Platt et al., 2001], the approach presented here allows us to track high-dimensional belief states, composed of thousands of hypotheses, using an accurate user-defined filter. However, the complexity of planning information effects is reduced by approximating the information value from a low-dimensional subspace of the belief space; five location hypotheses are used to test and evaluate this approach. Figure 1.5 shows a graphical representation of how to build trajectories that maximise the likelihood of localising a mug in a simple 2D case with two possible hypotheses for the object configuration.

In contrast to [Platt et al., 2001], this thesis uses such an approach to plan dexterous grasping trajectories that locate the object while simultaneously attempting to grasp it.

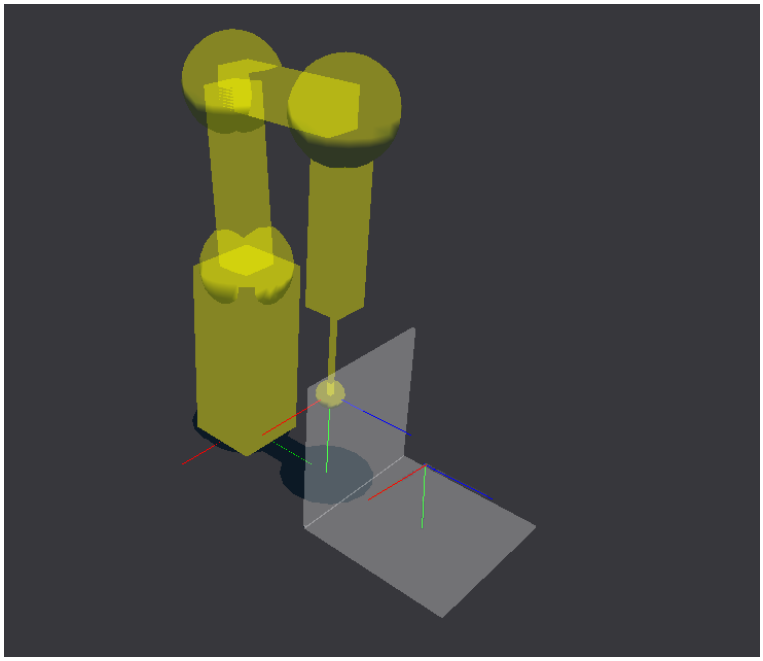
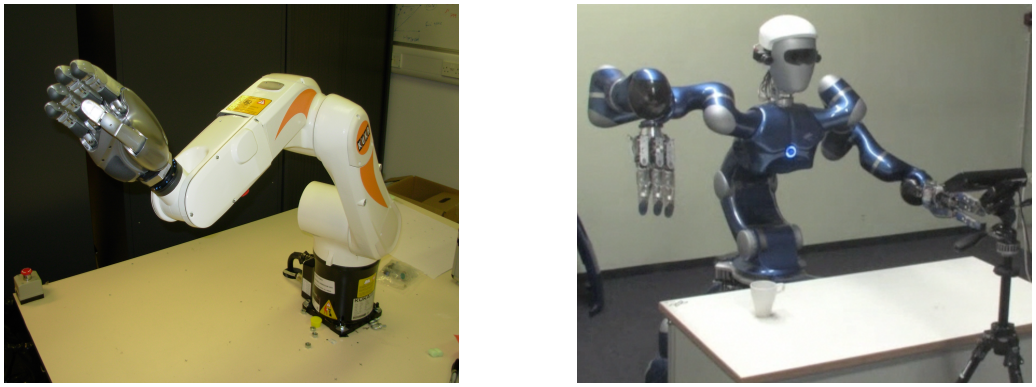


Figure 1.7: The push planning algorithm was tested using a simulation environment based on the NVIDIA PhysX physics engine, called Golem [Kopicki, 2010]. The test environment features a simulation model of a five axis manipulator (modelled after the Neuronics Katana 320 robot [Neuronics AG, 2004]), equipped with a single rigid finger with a spherical finger-tip, and a L-shape object called a "polyflap" [Sloman, 2006].

In addition, Platt et al. address the problem of locating the object to be grasped in 1D by using laser sensors prior to grasp execution, while in this thesis the robot is able to locate the object in 6D by using tactile exploration in the act of executing a reach-to-grasp trajectory. Figure 1.6 shows how a contact between the thumb of the robotic hand and the object to be grasped is used to refine the localisation of the object.

This work has been published in: (i) Workshop on Beyond Robot Grasping: Modern Approaches for Dynamic Manipulation. Intelligent Robots and Systems, 2012 [Zito et al., 2012a], (ii) Workshop on Manipulation with Uncertain Models, at Robotics: Science and Systems, 2013 [Zito et al., 2013a], and (iii) IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems 2013 [Zito et al., 2013b].



(a) (b)

Figure 1.8: Kuka arm KR 5 sixx R850 6DOF equipped with DLR-Hand II prototype of five-fingered hand 20DOF (left) and Rollin Justin robot developed at DLR (right).

1.4 Working scenarios

This section presents the virtual and real robot platforms used to develop the algorithms presented in this thesis.

1.4.1 Pushing scenario

The push planning algorithm was tested using a simulation environment based on the NVIDIA PhysX physics engine, called Golem [Kopicki, 2010]. The test environment features a simulation model of a five-axis manipulator (modelled after the Neuronics Katana 320 robot [Neuronics AG, 2004]), equipped with a single rigid finger with a spherical finger-tip. The choice of the object to be pushed was made to enable us to illustrate the potentiality of this approach for 3D objects that are free to flip, slide and rotate. Figure 1.7 shows the simulation of a Katana robot and the L-shape object (referred to as a "polyflap" [Sloman, 2006]).

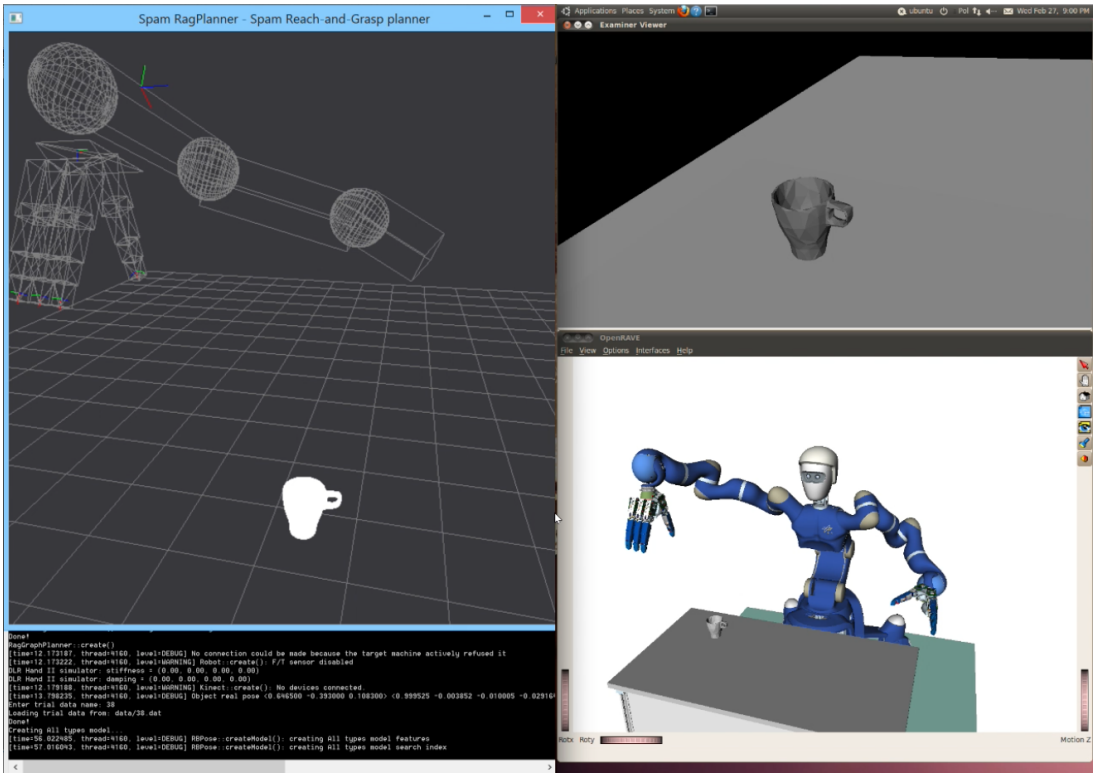


Figure 1.9: Several simulation environments have been developed. Left: a virtual environment based on the Golem framework [Kopicki, 2010], developed at UoB. Top right: a Bullet-based simulator developed at DLR, which is accurately calibrated to simulate hand-object physical interactions. Bottom right: OpenRave-based, [Rosen & Kuffner, 2008], Justin Simulator developed at DLR for control and visualisation of the entire robot. I have contributed to the development of the Golem framework.

1.4.2 Dexterous grasping scenario: pose uncertainty

The sequential re-planning approach for dexterous grasping under object-pose uncertainty has been developed for a Kuka arm KR 5 sixx R850 6DOF equipped with DLR-Hand II prototype of five-fingered hand 20DOF, and has also been integrated on the Rollin Justin robot developed at the DLR [Fuchs et al., 2009].

This approach is presented in Chap. 6. It has been developed and tested in several simulation environments: i) a Nvidia Physx-based simulator [Kopicki, 2010], and ii) OpenRave-based, [Rosen & Kuffner, 2008], Justin Simulator developed at DLR for con-

Figure 1.10: Boris with a set of graspable objects. The planning approach for information effects presented in this thesis has been developed for Boris' right arm equipped with DLR-Hand II prototype of five-fingered hand 20DOF.

trol and visualisation of the entire robot. Both simulators interface with a Bullet-based simulator developed at DLR, which is accurately calibrated to simulate hand-object physical interactions (Fig. 1.9). Several assumptions were made. First, I assumed that the real (tactile) sensing capability to retrieve a contact of Rollin Justin robot was exactly the same as in simulation, and that this sensibility was sufficient to palpate the object without perturbing the pose of the object itself. For proof of concept, I also tested this approach on the Rollin Justin robot (Fig. 1.8). However, in order to guarantee no movement after a contact, the object to be grasped needed to be glued on the desk. Therefore even though this approach converged to a force closure grasp, Rollin Justin could not lift up the object to prove the real stability of the grasp.

1.4.3 Dexterous grasping scenario: real world

The assumptions discussed in the previous section prevent the core method from working on a real robot. In order to bring us to a first convincing demonstration of the sequential re-planning approach on a real robot, several improvements and algorithmic extensions had to be done to cope with the relaxation of such assumptions.

The sequential re-planning approach has also been extended to perform on a real platform. Figure 1.10 shows Boris, the robotic platform developed at the University of Birmingham (UoB) with a set of everyday objects that Boris has to grasp.

1.5 Organisation

The flow of this thesis proceeds as follows:

Chapter 2 presents a literature survey on planning for robotic grasping and manipulation. First, the problem of grasping is decomposed into five subproblems: state estimation, grasp synthesis, reach to grasp trajectory and control. I present the work done for each subproblem which is related to the approach presented in this thesis. The chapter also presents research in neuroscience that supports the intuition that humans compensate for uncertainty in reach-to-grasp trajectories by modelling the problem as optimal decision making. Finally, a literature survey on robotic pushing will be presented.

Chapter 3 introduces the problem of path planning in continuous but deterministic domains. First, it will introduce the basic concepts from classical kinematics and then show how to efficiently plan collision-free trajectories for robot manipulators.

Chapter 4 presents a formulation for the problem of planning under uncertainty. Two kinds of uncertainty will be discussed, namely: uncertainty in physical effects and uncertainty in informational effects. The chapter will show the benefits and limitations of

this framework and explain why it is not possible to directly use decision-making algorithms to solve problems formulated in continuous state and action spaces. The chapter will also present hybrid methods which either extend motion planning techniques to cope with uncertainty or compute approximate solutions for decision making problems in continuous spaces.

Chapter 5 presents a two-level planner, for a robot manipulator equipped with a single finger, to solve the problem of moving an object to a desired location through pushing operations. The ability of this two-level approach is empirically evaluated in a simulated environment (Sec. 1.4.1).

Chapter 6 addresses the problem of planning dexterous reach-to-grasp trajectories for information effects. The main novelty is to simultaneously attempt to perform a task (here dexterous grasping) while gathering information (i.e. locating the object to be grasped), if necessary. Experimental results in a virtual environment (Sec. 1.4.2) will be presented that show how sequential re-planning is capable of converging to a grasp more robustly than single open-loop strategies. The chapter will also show that planning trajectories that maximise the information gain achieve successful grasps with fewer iterations.

Chapter 7 discusses the limitations of the sequential re-planning approach discussed in Chap. 6 and proposes extensions of the sequential re-planning algorithms that enable a robot to autonomously operate under object-pose uncertainty in a real scenario.

Chapter 8 will summarise the contributions of this thesis and will discuss the achievements, in terms of benefits and limitations, of the set of algorithms presented. Future work to overcome these limitations will also be discussed.

Chapter 2

Robotic manipulation and grasping

This chapter presents a literature survey on planning algorithms for robot grasping and manipulation under state and action effects uncertainty.

2.1 Organisation

The flow of this chapter proceeds as follows.

Section 2.2 provides an introduction to the problem of robotic grasping and to the set of sub-problems that need to be solved in order to achieve a grasp. These sub-problems are: estimation, grasp synthesis, grasp trajectory planning and grasp execution (control).

Section 2.3 introduces techniques to estimate the pose of an object. There has been an intensive use of techniques from computer vision to recover the state description of a grasping scenario directly from RGB-D data. This section will give a basic understanding of the procedures that will be presented in Chap. 6 for state estimation.

Section 2.4.2 presents an overview of techniques to synthesise a grasp, and to transfer a

grasp to *unknown* objects - by unknown I mean an object that the robot has never encountered before and for which no complete model is available. This will include a review of the metrics used to evaluate the "goodness" of a particular grasp. Although grasp synthesis is not the topic of this thesis, the algorithm presented in Chap. 6 uses force closure analysis (discussed in Sec. 2.4.1) to evaluate the quality of the grasps achieved by our proposed algorithms. In addition, Sec. 2.4.2 introduces the work of [Kopicki et al., 2014; 2015] for grasp synthesis. This method has been developed as part of the same framework used for developing this thesis. I have integrated this method in the set of algorithms that will be presented in Chap. 7.

Section 2.6 introduces the control problem associated with robotic grasping. In particular, we focus on the problem of interaction of rigid bodies and how to achieve compliance in order to cope with the forces generated by the contacts. Our proposed algorithms in Chap. 7 rely on the availability of an active compliant (see Sec. 2.6.2) controller to safety achieve contacts between a robotic hand and the target object.

Section 2.5 reviews the most relevant planning techniques that have been proposed to solve the problem of grasping in unstructured environments. This section presents the state-of-the-art planning techniques for grasping under state uncertainty. This section also summarises the conceptual differences between these approaches and the ones presented in Chap. 6 and 7.

In Sections 2.7 and 2.8, we introduce studies in neuroscience which provide possible explanations for the human ability to compensate for uncertainty during grasping. First, I compare human versus robot grasping in terms of the decomposition of the problem and the contrasting terminology from engineering and biology. I then provide a typological classification of human grasps. Experimental results suggest that the brain constructs such trajectories to minimise adjustments in the wrist orientation. Finally, the section presents research that investigates how humans compensate for uncertainty, due to noisy

sensing and imperfect sensory information, during the execution of reach to grasp movements. I will show in Chap. 6 that my experimental results are consistent with the conclusions in [Christopoulos & Schrater, 2009].

Section 2.9 introduces the problem of how to plan actions in the robot's configuration space to achieve a desired motion in the configuration space of the target object. These two spaces are related by complex physical laws that are hard to model analytically. Then the section presents a variety of techniques developed for planning pushing operations of objects. This will give the basic understanding my work presented in Chap. 5.

Finally, Sec. 2.10 summarises the contributions of this chapter to the thesis.

2.2 Robot grasping: Introduction to the problem

Traditional robot manipulators were designed to be used in industry and they are usually composed of an arm with a simple gripper attached as end-effector. These types of robots are very effective for tasks that require movement of large payloads and precise position control. However, such robots are effective only in structured environments, in which they can move safely with perfect knowledge of task-related characteristics, such as trajectories to execute, configurations and physical properties of objects they have to interact with, etc. In this case, the problem of controlling and planning movements is simplified by the lack of uncontrolled variables, such as shape and pose of the object to be manipulated. Another important disadvantage is that for a given gripper only a small class of objects can be stably manipulated. For example, a parallel gripper can efficiently grasp objects with parallel faces but it is rather inappropriate to grasp shapes with different geometrical properties.

In the past decades, dexterous grasping has been a central topic in robotics; nevertheless, robots have not reliable enough to pick up arbitrary objects in complex environments

with uncontrolled conditions. This is due to the intrinsic complexity of the problem that requires us to solve the following sub-problems:

1. *State estimation*, which addresses the problem of estimating some crucial quantities when the robot has no direct access to the states of the system, e.g. estimate the pose of the objects from visual data (see Sec. 2.3).

2. *Grasp synthesis*, which defines grasp-specific object-hand relations. This requires determining a set of contact points for the fingers on the surface of the object in order to achieve a stable grasp (see Sec. 2.4).

3. *Reach to grasp planning*, which addresses the problem of planning a trajectory of the arm to bring the hand in a pre-grasp position where it is possible to complete the grasp by closing the fingers (see Sec. 2.5).

4. *Grasp execution*, which focusses on the control problem associated with the execution of the planned trajectory. In grasping a robot has to interact with the environment through contacts, which requires applying forces and store energy (see Sec. 2.6).

5. *Finger closing strategy*, which achieves closure to a stable grasp given uncertainty and subject to the kinematic constraints of the hand. Typically this problem is not addressed explicitly. Once the robot's end effector is delivered to a pre-grasp configuration, in which the fingers cage the target object without making contacts, the closing strategy relies on the compliance properties of the end effector to apply sufficient forces to the object without damages.

In the following sections we will present the related work in more details. At the end of the chapter, table 2.2 summarises the problem of grasping under uncertainty at glance.

2.3 State estimation for robot grasping

Pose estimation of objects of interest is a critical component in many robotic applications, such as object grasping, bin picking, and self-localisation. In the recent years, the advent of fast and affordable range-imaging technologies, such as stereos camera processing, laser range scanners and time-of-flight cameras, have become crucial to improve sensing abilities of robots. Dense 3D data points are now available to produce accurate maps as geometric scene representations. However pose estimation for known objects in an unknown scene is still a hard problem to solve in computer vision. Natural scenes do not typically contain just the target object and different materials reflect laser rays in different ways, making some objects practically invisible.

A class of approaches used to estimate the pose of the object in robotic grasping relies on the *maximum likelihood estimate* (ML). Under the assumption that a model for the object to be grasped is available, in the sense of a dense point cloud model or a mesh model, a typical implementation for a robust and global estimator for computer vision problems is based on sampling subsets of perceived data points and computing parameter hypotheses based on the correspondences features between the data points and the underlying model. Both these approaches have been exploited in numerous variations, but mostly focussed on fitting shape models. In [Hillenbrand, 2008; Tuzel et al., 2005] object pose estimation problem is tackled by using mean-shift clustering in continuous parameter space. More recently in [Hillenbrand & Fuchs, 2011], the authors investigate the relative estimation accuracy and robustness of four variants of the pose clustering algorithm: hypotheses computed from subsets of range data points or from subsets of points with surface normals, each combined with clustering hypotheses in the canonical or consistent pose space. This work shows the superiority of the estimator, in the sense of robustness to data corruption and accuracy, when built upon subsets of points with surface normals and clustering in the consistent pose space. However small

errors in the pose estimate may lead to critical failures while attempting to grasp, i.e. unexpected contacts may damage the object or the hand itself.

Another class of approaches commonly used is to maintain as a state estimator a *belief state*, which is a probability distribution over the underlying states. In decision-making theory, there are many benefits of encoding states as probability distributions. One above all, it is the possibility to perform *a posteriori* computations in order to trade off the cost of executing information-gathering actions against the cost of executing inappropriate actions. However, this can be computationally expensive especially for high-dimensional, complex density functions. A popular choice is to constraint the belief space to Gaussian density functions. Unfortunately, in many problems in robotics there is no evidence that the belief space is well-represented by Gaussians.

Additionally, a mere extension of such formulations to non-Gaussian distribution yields to intractable planning problem due to the high-dimensionality of typical non-Gaussian parametrisations. A more recent class of approaches are based on approximating the belief space by sampling, this allows us to cope with multi-modal uncertainty. The *particle filter* (PF) is a *Sequential Monte-Carlo* (SMC) method that represents distributions by a cloud of particles. Each particle represents a possible hypothesis, which is a candidate pose of the object in the chosen configuration space (typically 2D or 3D). There are many examples of PFs used for state estimation in manipulation problems [Petrovskaya & Khatib, 2011], [Nikandrova et al., 2013] and [Platt et al., 2001]. Nevertheless, most of these methods sample an initial set of particles from an user-defined distribution attached to the *maximum a posteriori* estimate (MAP) obtained from vision, and often the uncertainty is limited to 2D. Yet, at my knowledge, there are no existing methods for robotic manipulation problems that can efficiently cope with non-Gaussian uncertainty in 6D (position and rotation) spaces.

In contrast, as a contribution to this thesis, I have implemented a similar method to the

pair-fitting model described in [Hillenbrand & Fuchs, 2011] to estimate the object pose from real data (further details in Chapters. 6 and 7). This estimation process is repeated hundreds or thousands of times to create a cloud of particles as belief state. This belief state represents non-Gaussian uncertainty in 6D, and it is used during planning to reason on the informational effects of a robot's actions.

2.4 Grasp synthesis and evaluation

The basic function of tools like multi-fingered hands is to grasp objects and possibly manipulate them by means of their fingers. A grasp is obtained by displacing the fingers on the object's surface and applying a set of forces to achieve some sort of *equilibrium* or *stability*, in which the grasped object is immobilised. Grasp synthesis refers to computational algorithms to construct stable grasps with robotic grippers or hands. The grasp should be constructed to ensure stability, task compatibility and adaptability to novel objects [Sahbani et al., 2012].

This section will first introduce the principal measures developed over the years to measure a grasp quality and then review the principal techniques to construct a grasp and to transfer it to novel objects.

2.4.1 Grasp stability

According to [Shimoga, 1996; Suarez et al., 2006], we classify the properties to measure a grasp quality in the following way:

- *Disturbance resistance*: a grasp can resist disturbances in any direction if and only if one of the following conditions is verified:
  - *Form-closure*: a pure kinematic analysis of the contact points which ensures

the immobility of the object.

  - *Force-closure*: when the fingers can produce appropriate forces to contrast any external disturbance in any direction. The fingers hence can apply appropriated force on the object to generate wrenches in any direction to compensate, up to a certain magnitude, wrenches generated by external forces.

- *Dexterity*: the kinematical relation between hand and the object enable the hand to move the object in any directions.

- *Equilibrium*: the resultant forces and torques applied to the object by both fingers and external forces are null. There is associated an optimisation problem to minimise the forces and torques applied by the fingers in order to guarantee the equilibrium point without damaging the object.

- *Stability*: the forces applied to compensate an external disturbance disappear in time after that the disturbance vanish, i.e. the grasp should return forces only if the object moves away from the equilibrium point. If at the equilibrium point no force is applied.

All of these properties have been investigated in grasp synthesis over the years, however the most frequently used are focussed on disturbance resistance and dexterity. In Chapter 6 three algorithms to plan reach-to-grasp trajectories are presented. These algorithms have been tested in simulation using a force-closure analysis to determine the quality of the grasp executed in each trial.

2.4.2 Grasp synthesis

Mainly two categories of approaches have been investigated to synthesise grasps [Sahbani et al., 2012]:

- *Analytical approaches*: construct contact points and hand pose that satisfy task constraints by a kinematic and dynamic formulation. However the number of parameters to be satisfied for a successful grasp grows with the number of fingers and the complexity of the object shape.

- *Empirical approaches*: construct grasps by mimicking human grasps to best conform to task constraints and object geometry. We refer to learning or classification methods to avoid the complexity of analytic approaches and task modelling.

Analytical approaches are posed as optimisation problems, and thus the computational effort arises when the grasp solution space is large. For example, if the object is modelled as a set of vertices, this kind of approaches have to search for all the possible combinations to find the optimal grasp, i.e. the one that maximises one of the grasp quality measures mentioned above. Yet, in task-oriented grasp planning, these techniques require a model for the particular task. In [Mishra, 1995] various grasp metrics for optimal force-closure grasps on 3D objects are extensively discussed with trade-off among the goodness of the a grasp, number of fingers required, the geometry of the object and complexity of the algorithm. All these metrics can be successfully used to construct optimal grasps for pick-and-place operations. However, even simple manipulation tasks in our every day life may go beyond the basic picking and placing actions.

On the other hand, empirical approaches are centred on observations. I distinguished two broad categories:

- *Learning by (from) demonstration*: the key idea is to have a robotic system that observes a particular grasp being performed and learning by demonstration. This can be achieved by either having a human *teacher* performing the grasp and learning a mapping between human hand shapes and artificial hand workspace, in terms of joint angles [Ekvall & Kragic, 2004; Fischer et al., 1998], hand shapes [Kyota et al., 2005], or corresponding wrench spaces [Argall et al., 2009], or simply human

teacher who guides the robot into a grasp learning directly the relations in the robot workspace.

- *Object-centric learning*: these approaches are based on the observation of relations between object's geometrical features and hand poses in order to compute grasps for a particular task.

In the recent years, there has been progress in learning task-oriented grasps that can be generalised to novel object's shapes. One way to achieve this is to exploit common local geometrical features in different objects in order to generalise grasps across object categories. In [Pelossof et al., 2004], the authors propose to use support vector machines (SVMs) to create a regression mapping between object shapes, grasp parameters and grasp quality. Once trained, this regression can be used to find the highest quality grasp associated with a new set of shape parameters. However the learning is limited to primitive shapes, as cylinders and spheres. Moreover, the authors used as quality measure the ability of the grasp to resists to the worst-case disturbance wrench, which generates optimal grasps mostly for pick-and-place operations.

Instead of hard-coding the mapping between shapes and grasp parameters, many researchers have focussed on learning such mapping from experimental data [Saxena et al., 2008; Detry, 2010; Kopicki et al., 2014]. The real challenge for such approaches is to learn this mapping from incomplete or erroneous data. In [Detry, 2010], the authors address the problem of associate a grasp to a partially perceived object form vision. They propose a method to learn a pre-shape configuration of the hand parametrised with respect to the object pose. The grasp is then achieved relying on compliance or tactile sensing.

Kopicki et al. propose an efficient method to learn dexterous grasp types (e.g. pinch, rim) from a single example that is able to generalise within and across object categories, and with full or partial shape information [Kopicki et al., 2014]. The main contribution

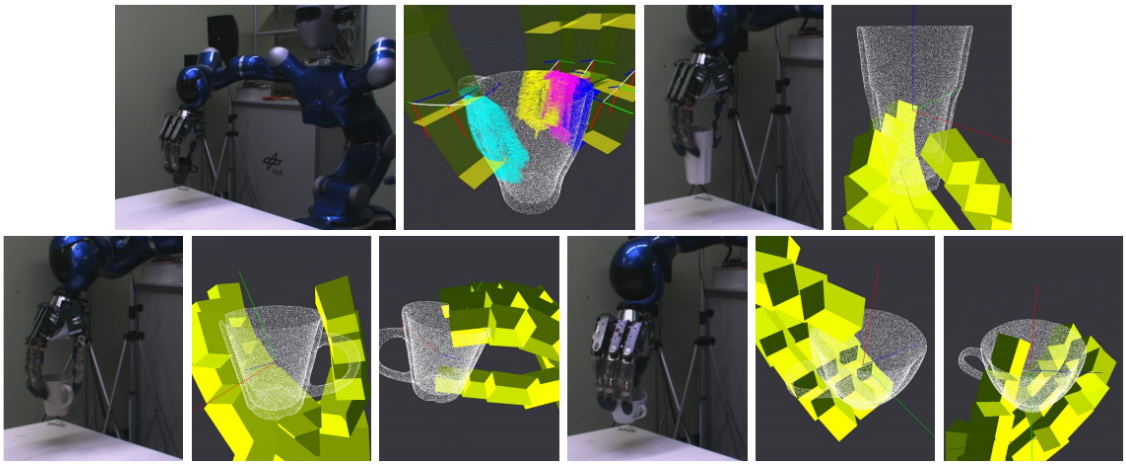


Figure 2.1: Mug top grasp. Top row - demonstration and learned contact model for mug 1; transferred grasp plus one additional grasp cluster for water glass. Bottom row - transferred grasps plus two additional grasp clusters for mugs 2 and 3. Reproduced from [Kopicki et al., 2014]

is to learn two types of models from the example grasp. The first model is what they call a *contact model*, which learns the relation between parts of the hand and the local surface of the object at the contact. This model is learnt for each link in the hand, and it is able to compute a set of locations on a new object surface in which similar contacts are generated. The second model learns a *hand configuration model* in order to combine the responses of each contact model at each finger's link in a way that a similar *global* configuration of the hand is produced. This approach is demonstrated to learn and transfer 6 grasp types to 31 different grasps on 18 objects, and it has been tested on two different dexterous hands: i) DLR-HIT Hand II and ii) DLR Hand II. Figure 2.1 shows the learnt contact for a mug and the transferred grasps on different mugs and a water glass. This method has been developed as part of the same framework used for developing this thesis. I have integrated this method to generate the target grasps for the planning techniques that will be presented in Chapter 7 .

2.5 Grasp planning

In this section, we discuss previous efforts in which the problem of robotic grasping is posed as a planning problem under state uncertainty. Section 4.4.1 presents a formulation for such problems, called POMDP. As we have already seen, the computational complexity to solve a POMDP depends on the number of states, actions and possible observations, hence POMDPs are often intractable except for small problems [Papadimitriou & Tsitsiklis, 1987; Madani et al., 1999]. Another issue is that if a robot has no direct access to the state of the system, an observational model is necessary to acquire information that can be used to produce an estimate of the state.

The use of sensory feedback has been intensively employed to solve many robotics problems, including pose estimation [Hsiao & Kaelbling, 2010][Corcoran & Platt, 2010], object recognition [Russell, 2000][Gorges et al., 2010][Chitta et al., 2010] and estimation of grasp quality [Dang et al., 2011][Bekiroglu et al., 2011]. However, most of the available sensors show characteristics such as drift, systematic errors and random noise. In addition, tactile sensors can provide only local information and are highly dependent on the structure of the manipulator. Recent approaches show a trend towards the probabilistic interpretation of tactile information, however the majority of existing approaches do not include the task goals in the probability frameworks, e.g. the work of [Hsiao & Kaelbling, 2010]. In pose estimation, for example, probing the object allows reducing pose uncertainty until a fixed threshold of confidence in the belief is reached. On the other hand, the convergence of the belief is not strictly necessary for different tasks such as grasping, in which for symmetric objects is not even attainable.

Furthermore, perception abilities have been combined with decision theory to investigate how to access information. This results in a *active perception* strategy in which actions are selected to maximise the gain of information relevant to a particular goal or task.

In [Lepora et al., 2013], a biomimetic finger is used to identify curvature values and locate in 2D an object by tapping on the object's surface. Bayesian belief integration is used to interpret the tactile observations and refine the estimated features (curvature and location). The proposed method, called Simultaneous Object Localization and IDentification (SOLID), combines sequential decision-making models with active sensorimotor perception to resemble various aspects of animal perception as further discussed in e.g. [Lepora & Gurney, 2012; Mitchinson et al., 2011].

From the literature, I identify two different approaches for sensor-based planning for robot grasping:

- *Global policies*: which address the problem of finding an approximate policy to enable a robot to act in any possible situations it may encounter. This is typically done by discretising the state, action, and observation spaces. However this kind of approach is mainly conceptual and typically evaluated only in simple scenarios.
- *Local policies*: which address the problem of finding an approximate solution for a small number of situations that the robot may encounter from a given initial state. In other words, these approaches condensate the search effort in the *reachable* belief space.

2.5.1 Belief planning: global policies

A probabilistic framework for sensor-based grasp planning was introduced by Hsiao et al. in [Hsiao et al., 2007], in which they present a simple blocks world problem, wherein a single finger can execute a small selection of actions (left, right, up, down) in response to a small set of possible sensory detections (contact or no contact below, right or left sides of finger). This simple problem is tractable for solving as a POMDP, producing optimal policies for guiding the finger towards a desired location on a 2D blocks world object

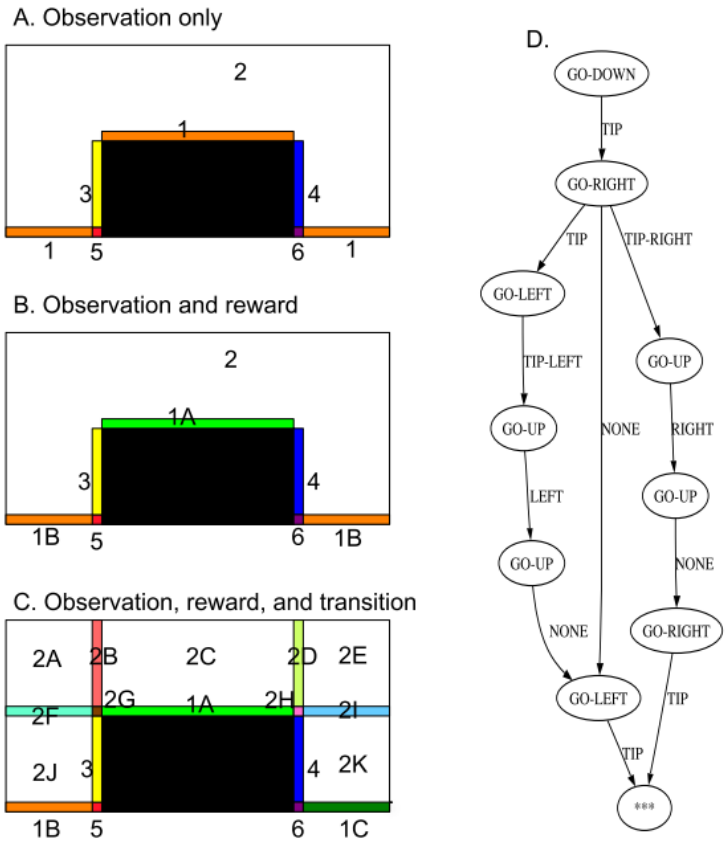


Figure 2.2: A. Observation partition; B. Observation and reward partition; C. Closed partition; D. Partial policy graph for robot starting in an unknown state above the table, with a deterministic transition and observation model. Reproduced from [Hsiao et al., 2007].

(Fig. 2.2). In contrast, I will present in Sec. 6.4.1 an observation model to cope with contacts for a dexterous robot hand in a continuous state space, and I use this model to refine the estimate of the object pose for non-Gaussian uncertainty in 6D. In addition, the approaches presented in this thesis do not require to compute an object-dependent set of information gain actions, but they will produce an optimised reach-to-grasp trajectory that simultaneously maximises the likelihood of gathering information while attempting to perform the target grasp.

Another example of real blocks-world has been realised by Toussaint in [Toussaint et al., 2010] by using a 14DOF Schunk arm and hand with tactile sensors and a stereo camera,

Figure 2.3: The robot has successfully put objects with green and red labels into separate piles using probabilistic inference for planning and control. Reproduced from [Toussaint et al., 2010]

the goal is to manipulate a set of objects on the table in a goal-oriented way. The key aspect of this work is a symbolic representation of states and actions which leads to high-level rule-based planning in relational domains. The work demonstrates the flexibility of approximate inference methods for control and trajectory optimisation on the motor level as well as high-level planning in an integrated real world, although sensors uncertainty is not taken into account (i.e. accidentally pushing objects off pile).

More recently, another probabilistic framework for sensor-based grasp planning has been presented in [Nikandrova et al., 2013]. Their approach philosophy differs from the one in this thesis, see Chapters 6 and 7, in the sense that the uncertainty in the pose of the object affects only the choice of the (final) grasp configuration of the robot - in terms of set of finger contacts and wrist pose - which maximises some criterions of stability. In contrast to the approach of this thesis, the authors do not reason about pose uncertainty (or about the information a trajectory would gain) while computing the reach-to-grasp trajectory. Their approach relies on the fact that the grasp chosen by the algorithm is the most robust in the face of novelty. To achieve so, they have proposed two strategies to synthesise the grasp: either based on i) stability maximisation

or ii) entropy minimisation. The former approach is implemented as a trial and error procedure in which tactile measurements, at the end of the grasp trajectory, are used to update the belief state - represented as a particle filter - and then the robot is moved to a fix position at the beginning of each iteration. The latter approach selects the most informative grasp to reduce entropy and fasten the convergence to a stable grasp. However this requires a *a posteriori* analysis which is expensive as the belief space grows. Both approaches have been evaluated in simulation with a proof of concept on a real robotic platform, a Melfa RV-3SB 6DoFs arm and a Robotic 3-finger hand. However the experimental setup consists of a simplify 2D problem in which a 2D Gaussian noise is manually added to the object estimate. In contrast, the methods presented in Chapters 6 and 7 can cope with non-Gaussian uncertainty in 6D generated directly from the visual sensory data.

2.5.2 Belief planning: local policies

Hsiao later addressed real-world grasping with an arm and three-finger Barrett hand equipped with tactile fingertip sensors [Hsiao & Kaelbling, 2010]. Because the space of possible actions for such a robot is enormous - especially compared to the single finger and 2D blocks-world problem of [Hsiao et al., 2007] - in order to solve the problem as a POMDP, the authors restrict the robot's choice of actions to executing a small number of pre-programmed reach-to-grasp motions, described relative to the pose of the target object. Thus the POMDP method can tractably be used to select between this small number of actions, in response to tactile contacts detected by the three fingertip sensors during the previous action. Thus various actions are sequentially selected, with successive refinements of the object pose collected from sensing during each action, until eventually one of the actions achieves a successful grasp.

The same authors in [Hsiao et al., 2011] extended their probabilistic framework to select

Figure 2.4: Illustration of the grasping problem, (a). The robot must localize the pose and dimensions of the boxes using the laser scanner mounted on the left wrist. This is relatively easy when the boxes are separated as in (b) but hard when the boxes are pressed together as in (c). Reproduced from [Platt et al., 2001].

grasps more robustly with respect to erroneous object recognition and motion error due to incorrect calibration. They claim that Bayesian formulation, in which the expected success is maximised, leads to more robust approaches than just executing the most likely action. Other authors in [Weisz & Allen, 2012] use a similar formulation by integrating a grasp quality metric over a uniform uncertainty in the object pose to rank a set of pre-computed grasps associated to the object.

Petrovskaya also investigated the use of tactile information, derived from collisions of an end effector with an object, to refine localisation estimates for that object [Petrovskaya & Khatib, 2011]. Petrovskaya represents the location of an object to be grasped, as a collection of particles forming a distribution, and this distribution is refined by successive manipulator contacts with the object. However, the work is limited in that Petrovskaya does not address the problem of planning manipulator trajectories to achieve these contacts. Instead, the author simply begins with an assumption that a selection of collisions will occur, and then shows how to use such collisions to update an object location distribution. In contrast, this thesis presents an algorithm for planning reach-to-grasp trajectories that actively gather information about the object location, if it should turn out not be located at its expected position.

Platt et al. propose a formalisation of the problem of grasping a belief space control problem. In other world, this formalisation allows them to find a sequence of motor control actions for grasping while bounding the probability of failure. This approach has been applied to a real scenario on which the robot needs to localise and grasp a box (Fig. 2.4). Two boxes of unknown dimension are presented to the robot. This situation is challenging because the displacement of the two boxes might make harder the problem of identify the exact pose of the boxes themselves. In their example, the robot is equipped with two paddles and it has a pre-programmed "lift" function. The robot localises objects using a laser sensor mounted on its left wrist. The boxes, however, are assumed to be placed at a known height and thus the robot has uncertainty only in one dimension. In contrast, this thesis extends this hypothesis-based approach in several ways: (i) plans trajectories for dexterous manipulators; (ii) copes with object pose uncertainty in 6D; (iii) does not rely on a pre-programmed grasp action.

2.6 Grasp execution

In grasping and manipulation the robot interacts with its environment through contacts, which means that there is an exchange of forces between robot's end effector and the environment. If the robot is composed of rigid links and stiff actuators, it is possible to move it to a desired position or track its position along a trajectory with high accuracy. However, the actuator will attempt to reach the desired position even in the presence of external forces, this may cause the robot to knock over objects or, in the worst case scenario if the external forces are not within the limits of the actuator, may damage the robot itself.

On the other hand, a compliant actuator allows deviation from its equilibrium position. The equilibrium position is reached when the actuator generates zero forces or torques.

If external forces are applied to the actuator, it will act as a spring system, which allows us to store energy.

In many everyday situations, richer strategies could be attainable by exploiting contacts as, for example, in grasping small objects (like a pen or screwdriver) on a planar surface. In this case, safety landing fingers on the supporting surface before sliding into contact with the object would be a feasible solution. This approach also mimics our understanding of human behaviour while interacting with the environment.

For the scope of this thesis I only distinguish two approaches for designing compliance systems: (1) the compliance is of mechanical nature or (2) a feedback control is implemented to simulate a spring behaviour when external forces are detected. More recently, other hybrid approaches were developed, e.g. Variable stiffness actuators (VSAs) [Tonietti et al., 2005], which optimise performance while intrinsically guaranteeing safety. Nevertheless, hybrid approaches are beyond the aim of this thesis due to the fact that they are designed to be performant in highly dynamic movements, such as throwing or catching, while through this thesis I assume a quasi-static environment.

In the following sections, we briefly overview both approaches.

2.6.1 Mechanical

Passive compliance actuators are built with elastic or flexible materials, thus compliant hands act as a spring that can store energy. A compliant device can resist to reasonable shocks and collisions with rigid obstacles, however the amount of compliance is typically fixed and determined by the design of the device.

This approach yields to devices that are simpler, cheaper and more robust with respect to collisions. However such devices typically are under-actuated which simplifies the controller at the cost of a lack of controllability, which effects negatively the proprioceptive

(a) **(b)**

Figure 2.5: Two examples of compliant hands: (a) RBO Hand 2 and (b) PISA-IIT Soft Hand. RBO Hand 2 has 7 DoFs, it is pneumatically actuated and made of silicone rubber, polyester fibers, and a polyamide scaffold. Reproduced by [Deimel & Brock, 2014]. PISA-IIT Soft Hand is an anthropomorphic hand prototype with 19 DoFs and only a tendon (which works as an actuator) distributes the motion of the joints. The dimensions of this prototype resembles the ones of an adult human. Reproduced by [Catalano et al., 2014]

tion and reduces the postures achievable. Nevertheless, one of the main advantage of using passive compliant hands in grasping is that the hand is capable of adapting to the object's shape during the grasp execution, which allows us to develop grasping strategies for unknown objects with no need to model such an uncertainty.

Deimel and Brock in [Deimel & Brock, 2014] have presented the RBO Hand 2, which is a compliant hand controlled via a pneumatic actuator. The authors have shown to perform reliable pick-and-place operations under sensing, model and actuation uncertainty. Similar performances have been shown for tendon-driven hands, such as THE Second Hand [Grioli et al., 2012], the Pisa-IIT Soft Hand [Catalano et al., 2014] and the SDM hand [Dollar & Howe, 2008]. Nevertheless, the lack of controllability of these devices is their main limitation in everyday situations in which even simple manipulative tasks may go beyond the basic picking and placing objects.

2.6.2 Feedback control

Active compliance actuators do not have physical compliant components, thus they mimic compliance adapting the stiffness of the actuator. The active compliance actuator is not able to store energy *per se* and, due to physical limits in the velocity with which the controller can react to external forces, no shock can be absorbed by the device.

Active compliance grippers or hands are typically fully actuated and the stiffness of its joints is controlled by a separate motor. Several designs have been proposed over the years. Some hands are all modular, like the DLR Hand I and its successor the DLR Hand II [Butterfass et al., 1998; 2004], where all the fingers have equal length and kinematics. The reduction in the complexity and costs of these type of hands comes at the price of a limitation in terms of performance. In fact, in primates the thumb is different from the rest of the fingers in terms of length, kinematics and strength, which is essential for performing both fine manipulation and power grasp. Other designs have been proposed to overcome such problems. The Shadow Hand, for example, is an anthropomorphic hand with 18 DoFs and two additionally DoFs in the wrist [ShadowRobotCompany, 2003]. The thumb and the little finger have 5 DoFs while the rest of the fingers have 4 DoFs.

2.7 Human vs. robot grasping

The hand is both a sensory and a motor organ [Novak & Hermsdörfer, 2009]. In Sherrington's terms, the sensory skills of the hand can be viewed as the fovea of the somesthetic system, to some degree the hand plays the same role of the centre of the retina in the visual system. The hand is used to explore the haptic world by touching, grasping and manipulating objects as eye movements are used to explore the visual world by changing the fixation point. Though sensory and motor functions of the hand are not disjointed,

Table 2.1: Comparison of stages in human vs. robot grasping. The table shows in the top rows the identified stages of kinematic planning. There is evidence that the human brain plans for the displacement of the fingers as well as for the applied forces. The bottom rows of the table show the stages involved in the execution of the planned trajectory. Experimental results show that humans adapt the grip-width during the reaching movements, whilst in robotics typically the reaching movements aim to transport the hand to the pre-shape pose.

HUMAN GRASPING	ROBOT GRASPING	
Digit position + Force setting	Grasp synthesis Planning reach to grasp	Kinematic
Hand preshape Hand transport Grasp	Grasp execution Finger closure	Dynamic

they are complementary. Palm and fingers movements contribute to explore and perceive objects whilst the sensor input from skin receptors contributes to control the hand movements. In evolutionary terms, the hand in primates has moved away from locomotion purposes developing a powerful and opposable thumb. In the same way, the primate brain shows extended motor and sensory cortex areas to control and process information related to the hand. However, "*despite the broad acknowledged importance in primate behaviour and cognition, many aspects of hand function have barely received attention until recently*" [Novak & Hermsdörfer, 2009].

Of particular interest, Klatzky and Lederman argue "that knowledge of biological tactile/haptic systems potentially provides many new avenues in the design of autonomous and teleoperated sensor-based robotic systems that incorporate touch (with or without vision). The work on haptic exploration is particularly relevant to the performance of complex robotic perceptual and manipulatory tasks in highly unstructured environments." [Lederman & Klatzky, 1993]. To a great extend the same authors have investigated the sense of touch in humans and its importance in everyday tasks (see e.g. [Klatzky

et al., 1985]), developing a systematic research programme focussed on analysing how fast and accurate blindfolded adults are on haptic perception and recognition of everyday objects [Lederman & Klatzky, 1993].

Neurophysiological aspects of motor control in anaesthetised animals have extensively been studied since the time of Sherrington while little attention has been given to reach-to-grasp movements, mainly because the high dimensionality of the action space for primate arm and hand (up to 25 degrees of freedom) makes hard to develop a complete description of these movements. Many researchers hypothesised that human efficiency in selecting grasps derives from our ability of learning from experience how to reduce a large search space. Therefore, most of human grasps would derive from few postures; an attempt to classify such basic postures yield to the concept of *grasp taxonomy*.

A first breakthrough was introduced by Napier in [Napier, 1956] with his functional description of prehensile movements of the hand. Napier's intuition claims that stability of prehension is a prerequisite for further manipulative activities and it can be achieved in two ways: "*(1) the object may be held in a clamp formed by the partly flexed fingers and the palm, counter pressure being applied by the thumb lying more or less in the plane of the palm. This is referred to as the power grip. (2) The object may be pinched between the flexor aspects of the fingers and the opposing thumb. This is called the precision grip*". These two patterns can be used either separately or in combination to describe all possible prehensile activities in humans. Napier, nevertheless, describes only the final posture of the hand during the hand/object interaction but he does not address the dynamic aspects of how those postures can be achieved. Several researchers have attempted to extend Napier's work. One of the most frequently cited is Cutkosky's grasp taxonomy [Cutkosky, 1989] which includes several basic postures according to the finger shape, such as *prismatic* and *circular* grasps. More recently, Zheng et al. investigated the usage frequency of the grasp types from Cutkosky's grasp taxonomy for common

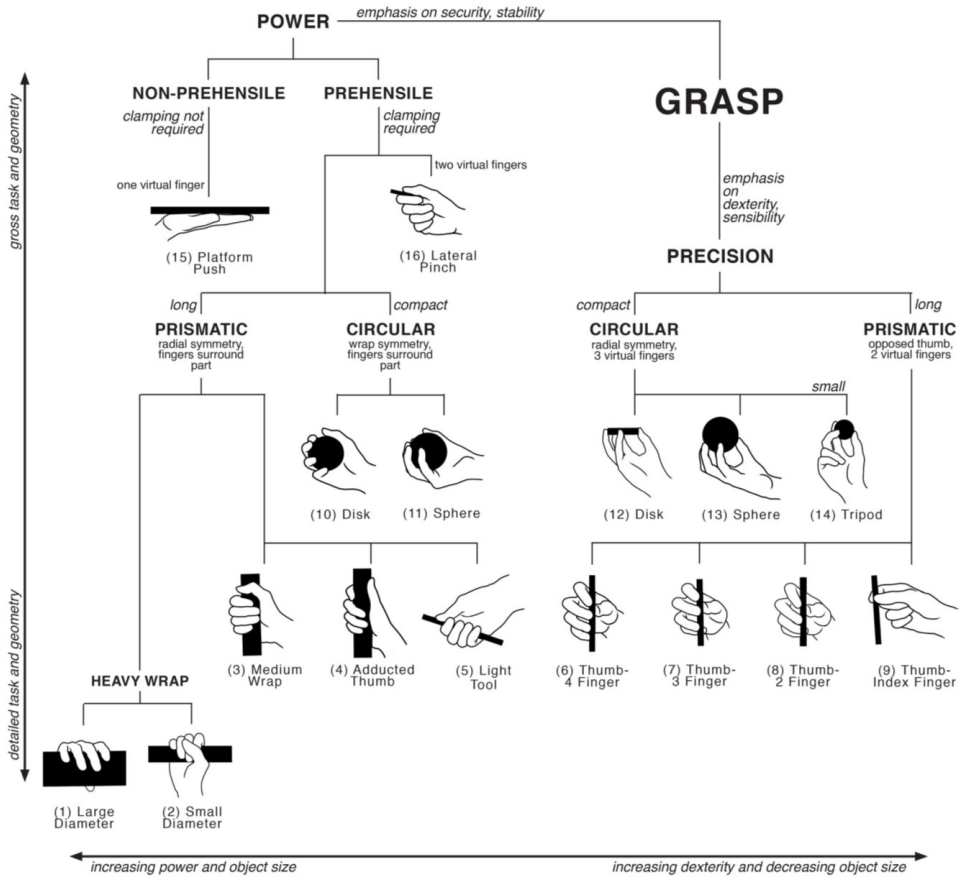


Figure 2.6: Graphical illustration of the extended Cutkosky's grasp taxonomy for daily household objects. Reproduced from [Zheng et al., 2011].

daily household objects in order to design better robotic or prostatic hands [Zheng et al., 2011] (see Fig. 2.6). In contrast, Ciocarlie et al. applied Principal Component Analysis (PCA) to the configuration space of robotic hands to form a low-dimensionality space for grasp postures, termed *eigengrasp*. This framework can be used to simplify the search for the task of grasp synthesis (see Sec. 2.4.2) even for complex hand designs. For a known object, according to its geometry, the eigengrasps are computed and an optimisation is performed to find a form closure grasp (see Sec. 2.4.1) as a combination of eigengrasps.

Reach-to-grasp movements, which include the transportation of the hand to the loca-

tion of the object, become a central topic in neuro-physiological studies in the 1970s. Kuypers, Mountcastle, and Faugier-Grimaud led extensive and comprehensive studies on brain mapping in visually directed reaching and manipulation activities on monkeys. Successively, various researchers have split prehension into two major components: (1) movements to reach for an object and (2) manipulative actions. Jeannerod at al. in [Jeannerod & Biguer, 1982] proposed a subdivision in "visuomotor channels" such that an "object" channel would encode intrinsic properties of the object to be manipulated to shape the hand accordingly to the size, shape and orientation of the object. In parallel, the "space" channel would encode the spatial relationship between the object and the body in order to reach for it.

A more complete description of these two components was proposed during the 1980s. A breakthrough was obtained by reducing the degree of freedom taken into analysis during reach-to-grasp trajectories. In [Jeannerod, 1981], subjects were instructed to reach, grasp and lift an object at normal velocity rate using a precision grip with only thumb and index finger. These experiments yielded to describe reach movements as curved trajectories of the hand from the starting position of the hand to the object location, also showing that the starting hand position was combined with the object orientation in order to minimise adjustments in the wrist orientation. The subjects were filmed during the experiments at a rate of 50 frames per second and manually the positions of three landmarks were extracted from the films (wrist, thumb tip and index tip). A systematic analysis, still in use nowadays, was proposed, such as movement time (MT), measured in number of frames from the first detectable wrist movement to the first detectable movement of the object, time to peak acceleration (TPA), time to peak velocity (TPV), time to peak deceleration (TPD) and maximum grip aperture (MGA).

The MGA shows that there is a strong correlation between object properties (e.g. shape, size) and the displacement of the fingers along the reaching movements. This correlation

provides evidence of the existence of an anticipatory computation in the brain, by the visual system. Other studies have systematically varied the size of the object over trials which shows a linear correlation between MGA and object size (e.g. [Marteniuk et al., 1990]).

The work in [Jeannerod & Biguer, 1982] investigated another question: if the prehension movement is split into two components, how these components are synchronised with each other? The author studied the relative timing between the wrist displacement and the finger closure exploiting the simple intuition that at the moment of fingers closure the wrist velocity must drop to zero otherwise the grasp might be jeopardised. However the temporal correlation between the transportation component and the fingers closure seems to be little affected by the lack of visual information during reach-to-grasp movements.

2.8 Human grasping as a decision making problem

More recently, many researchers attempted to explain human behaviours as an optimal decision making problem in the face of uncertainty. Wolpert and his colleagues have shown that the sensorimotor system in humans employs state estimations of both target and limbs which are coherent with optimal Bayesian framework [Körding & Wolpert, 2004]. Bayesian state estimation and decision making have been combined to in a large number of ways to derive models for optimal motor control (e.g [Körding & Wolpert, 2006], [Landy & Wolpert, 1012]). Nevertheless, none of these models exhaustively explain which criteria our motor system might use.

Schrater and his colleagues have investigated how humans compensate for uncertainty due to noisy motor commands and imperfect sensory information during the execution of reach-to-grasp movements. In [Christopoulos & Schrater, 2009], they induce direc-

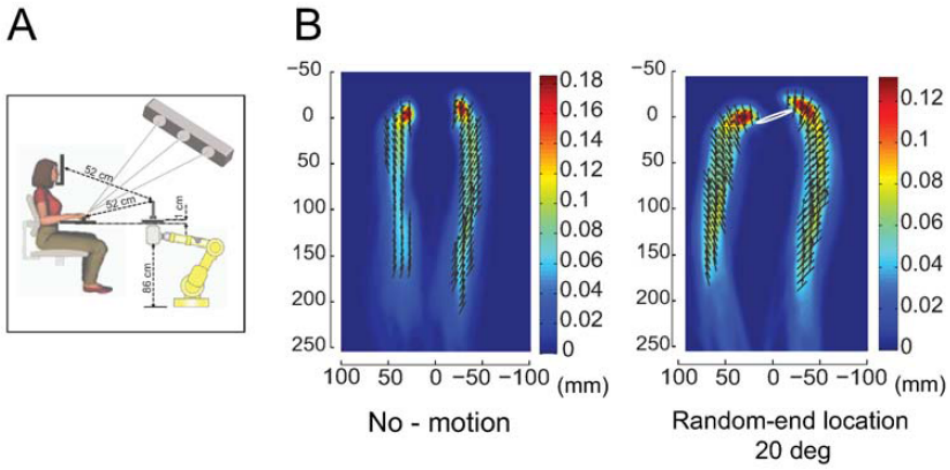


Figure 2.7: Graphical illustration of grasp analysis with directional position uncertainty. (A) Illustrates the experimental apparatus where participants grasped and lifted a cylindrical object with position uncertainty. (B) Examples of grasp trajectories for thumb and index finger without uncertainty (left: no-motion of the cylindric object) and with uncertainty (right: random-end location). A superimposed density map shows the probability of a trajectory passing through each spatial location, where blue and red indicate zero and high probability, respectively. Reproduced from [Christopoulos & Schrater, 2009].

tional object-position uncertainty on the object to be grasped in order to investigate the compensation strategies adopted by humans in such tasks. The subjects were instructed to reach quickly a cylindrical object mounted on a robot arm and lift it. The object was visible through liquid crystal glasses that were shut before the participant could start the reaching movement. The apparatus used for the experiments is shown in Fig. 2.7. Three conditions were presented to the participants:

- A *no motion* condition in which the cylinder was stationary throughout the reach.
- A *fixed-end location* condition in which the robot arm moves visibly the object in randomly drawn positions, after the object is occluded by shutting the glasses, the object is moved to a fixed location.
- A *random-end location* condition in which the robot arm moves visibly the object in randomly drawn positions, after the object is occluded by shutting the glasses, the object is moved to a new random location.

The main contribution of the work in [Christopoulos & Schrater, 2009] is that they use normative prediction to evaluate the benefits of uncertainty compensation, based on the hypothesis that participants prefer to generate force-closure grasps at first contact. Participants, in order to be time efficient, tended to modify their approach along the direction of maximum uncertainty and increase peak grip-width (MGA). Interesting is that the fixed-end location condition affects only the grip-width strategy, whilst the random-end location condition triggers a compensation in both gip-width and approach direction. Thus "the results suggest that the sensorimotor system cannot ignore the cylinder motion even when it is uninformative".

2.9 Robot manipulation planning

Robotic manipulation poses an interesting and challenging planning problem, in that actions must be planned for a robot manipulator, but the desired motions of interest belong to the manipulated object. As we have seen, this touches on the general problem of how to plan actions in one space (e.g. the joint space of an arm, see Sec. 3.2.3) which have an effect on another space (e.g. the configuration space of a manipulated object), where the relationship between the two spaces may be complex.

A contribution of this thesis is to address the problem of planning sequence of pushing operations in 3D domains. It is non-trivial to build accurate forwards models (predicting the motions that will result from an action) for pushed objects, especially 3D objects which are free to tip, topple and rotate in complex ways, as well as slide across a surface. The corresponding inverse models (computing a push to cause a desired motion) may be intractable or otherwise unavailable.

As we will discuss in Chap. 3, there are well-developed paradigms for solving motion planning problems for a robot that is planning how to move itself, relative to some

obstacles (see [LaValle, 2006] for a broad survey). Typically these rely on the availability of explicit inverse kinematics models to enable the robot to reach the desired nodes on a graph-like path structure requested by the motion planner. In contrast, one of the main contributions of this thesis is to develop algorithms for planning sequences of push manipulations for which such inverse models are unavailable.

In this section, we will discuss first how to build forward models for planning and then previous attempts to solve the problem of finding a sequence of pushing operations will be reviewed.

2.9.1 Forward models

When the hand is in contact with the object to be manipulated, in order to achieve the task, either grasping or manipulation, we necessitate the ability to predict the behaviour of the object under the manipulative actions. For this reason, it is necessary to have a forward model (FM) which describes how our actions affect the environment. In addiction, the choice of the planning formalism is strictly correlated with the forward model chosen. Two are the possible options: (1) physics engines and (2) learned predictors [Kopicki, 2010].

In *physics engines*, objects are defined by geometric primitives and their motions are predicted in terms of rigid body transformations using the law of physics. Physics Engine subsequently detects all collisions as sets of contacts and modify the movement of simulated bodies using contact resolution methods. Unfortunately, none of these methods are without drawbacks. Friction and restitution are particularly difficult to model and frequently lead to situations that violate the law of energy conservation. Furthermore, the order in which contacts are resolved is critical and has a great influence on the predicted motion. On the other hand, continuous methods are relatively insensitive to the contact resolution order since they explicitly handle deformation during a single

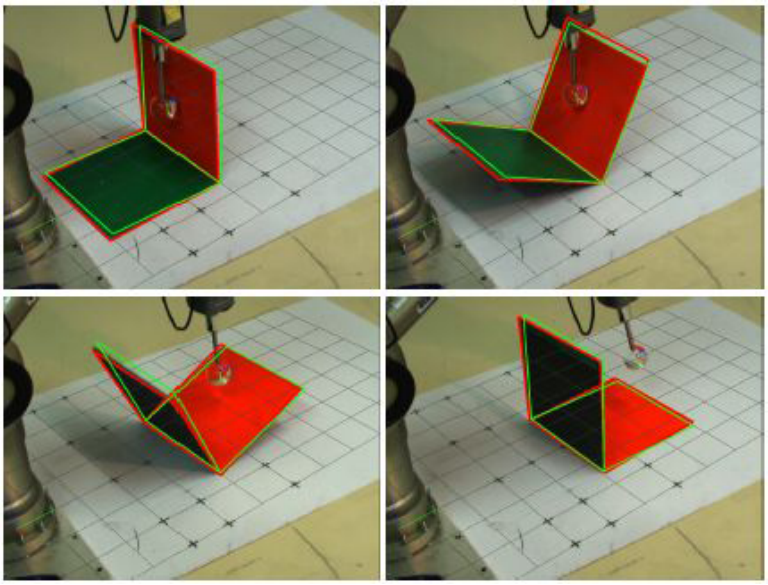


Figure 2.8: The example shows the interaction between a 5-axis Katana robotic manipulator and an L-shape object, called a polyflap. The green wire frame denotes the prediction whilst the red wire frame denotes the visual tracking. Reproduced from [Kopicki, 2010].

contact. These models, however, are expensive and appropriate parameters of the model can be difficult to obtain in practice.

Learned predictors, instead, are able to encode physics information without explicitly representing physics knowledge. Prediction is already used in robotic manipulation, in particular when it involves planning and interaction with the real world. As the real world is governed by laws of physics, most previous robotic approaches use either physics simulators or other kinds of physics-derived parametric models. This has led some researchers to suggest the abandonment of analytic approaches in some cases: "Clearly analytical solutions to the forward dynamics problem are impossible except in the simplest of cases, so simulation-based solutions are the only option" [Cappelleri et al., 2006].

In the work presented in [Kopicki, 2010], the author shows that various geometric relations between parts of objects can be represented as statistically independent *shape/-*

contact experts distributions, and when used in *products of experts* allow us to generalise over shape and applied actions, as well as to effectively learn in high dimensional space. In other words, the study in [Kopicki, 2010] is about predicting what can happen to objects when they are manipulated by an agent, for example, a robot. Although in this study the author considers only simple pushing manipulation by a robot, the findings are more generally applicable to predicting more complex interactions. In this study, the author explores alternative approaches to using physics engines, including learning methods. Specifically: (1) how forward models can be learned with a high accuracy and generalised to previously encountered objects and actions; and (2) explore a simplified physics approach which can be combined with the prediction learning approach.

Learning of forward models is one of the most promising alternatives which could avoid many of the problems mentioned above since it does not need to refer to any fixed model of the world, and thus avoids the limitations of such models. Nevertheless, this thesis presents an alternative approach (discussed in Chap. 5), in which simple physics principles are used to infer the likelihood of candidate object motions, without the need for learning.

2.9.2 Planning sequence of pushing operations

Pushing operations are encountered frequently in robotics, but have received comparatively little attention in the research community. As we have seen, pushing is the most primitive kind of manipulation, but is challenging in that the effect of a push on the object motion is complex and can be hard to predict. This requires a planning task which is therefore non-trivial, and this is what is investigated in Chapter 5.

Pushing is also important in that pushing is a component of more complex tasks such as grasping [Mason, 2001; Dogar & Srinivasa, 2010]. When a two fingered gripper or a multi-fingered hand approaches a grasp configuration, uncertainty means that one finger

will typically contact the object before the others, resulting in a single finger pushing phase before a stable grasp is achieved. In-hand dexterous manipulation motions are essentially the (non-linear) superposition of the effects on a object of pushing motions due to each of the contacting fingers. More recently, Dogar and Snirivasa have developed a quasi-static tool that can be used for analysis and simulation of push-grasping actions for dexterous hands [Dogar & Srinivasa, 2010]. This tool is used to efficiently derive motion planning trajectories for stable push-grasping operations under object pose uncertainty in clutter scenarios.

Mason, was the first to identify pushing operations as fundamental to manipulation, especially grasping [Mason, 1982]. Mason developed a detailed analysis of the mechanics of pushed, sliding objects and determined conditions required for various 2D motions of a pushed object. In [Peshkin & Sanderson, 1988] the authors attempted to put quantitative bounds on the rate at which these predicted motions occur. Lynch developed a method for finding the set of all possible motions of a sliding object, in response to an applied push [Lynch, 1992].

More recently, the work of [Cappelleri et al., 2006] has experimented with push manipulation of 2D sliding objects at the micro-manipulation scale. The study presented in [Cappelleri et al., 2006] describes a test-bed for planar micro manipulation tasks and a framework for planning based on quasi-static models of mechanical systems with frictional contacts. It shows how planar peg-in-the-hole assembly tasks can be designed using randomized motion planning techniques with Mason's models for quasi-static manipulation [Mason, 1982].

The goal is to use simulation and motion planning tools to design open-loop manipulation plans that rely only on an estimate of initial position and orientation. An example is shown in Figure 2.9. They use Mason's quasi-static models for manipulation of planar parts with surface friction. A method for 3D simulation is adapted to solve the "2.5-

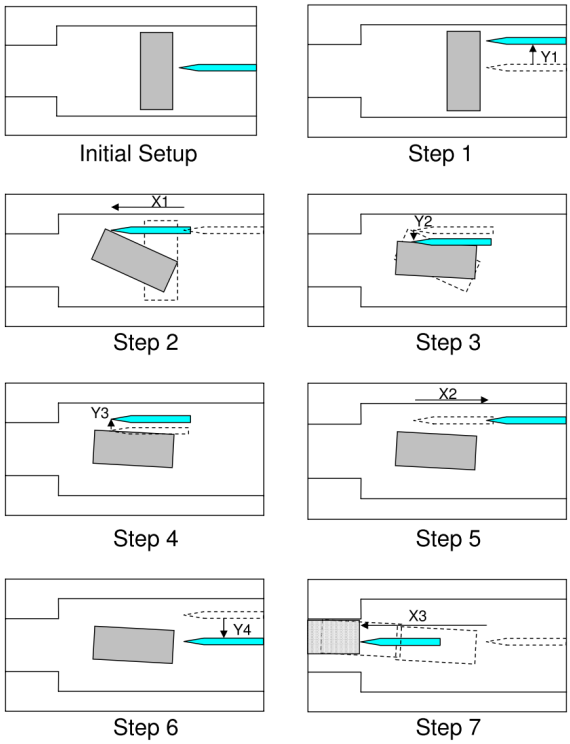


Figure 2.9: Open-loop motion plan for assembly. Reproduced from [Cappelleri et al., 2006].

dimensional“ problem with surface friction. An application of the Rapidly Exploring Random Tree (RRT) algorithm with modifications for dynamic systems is used to solve the peg-in-hole insertion task. However, all of this work is restricted to planar sliding motions of what are effectively 2D objects. There is little literature addressing the more complex problems of push manipulations on real 3D bodies, which are free to tip, topple or roll, as well as slide, and for which achieving a desired tipping or toppling effect on the object may actually be a critical objective of the manipulative operation.

A 3D pushing scenario for mobile robots has been investigated in [Meriçli et al., 2014]. In this case, the authors proposed to collect experimental data of a mobile robot that interacts by pushing with passively-rolling caster wheels. The data is collected by demonstration, the robot is tele-operated by a joystick, or “self-learning”, where the robot applies a determinate push several times with different duration times. The authors have demon-

strated in simulation that a robot is able to construct plans to move a passively-rolling caster wheels from a starting pose to a desired one by using the data collected as building blocks. In addition, empirical results have shown the validity of such an approach in a real scenario.

2.10 Conclusion

This chapter introduced the problem of robotic manipulation in unstructured environments. The lack of "structure" is the main source of uncertainty, in the sense that a complete model of the environment is not available or achievable, hence the necessity of developing robust strategies to act in partial ignorance.

We started analysing the problem of robotic grasping under object pose uncertainty. We decomposed this problem as follows: state estimation, grasp synthesis, grasp planning and control (see Tab 2.2). In Sec. 2.3, we discussed the advantages of encoding states as probability distributions instead of relying on the ML estimation. In fact, if the object is not in its estimate pose an open-loop grasp trajectory may lead to failures. In contrast, maintaining a density over the pose of the object yields to probabilistic strategies that are capable of maximising the likelihood of success as well as gathering information if necessary, as in the work of [Hsiao et al., 2011]. In particular, the section highlighted the benefits of using a representation of the belief space such as a PF since many robotic problems have typically multi-modal uncertainty. In contrast to other work, e.g. as in [Nikandrova et al., 2013], in which the initial set of particles is sampled from a user-defined distribution, the approach of this thesis is to estimate a set of hypotheses (or particles) in a PF fashion from real RGB-D data, where each hypothesis is the result of a ML estimation (see for further details Chapters 6 and 7).

Section 2.4 discussed techniques to construct grasps with robotic grippers or hands,

and how to evaluate the grasp stability. We distinguished between: i) analytical or ii) empirical approaches (discussed in more details in Sec. 2.4.2). The former is typically associated with an optimisation problem, and thus, the computational effort grows with the dimensionality of the grasp solution space that is related to the number of fingers and number of contact points. On the other hand, empirical approaches learn a mapping between object's parameters and grasp space.

Section 2.5 we presented a literature review of probabilistic grasp planning techniques for grasping under object-pose uncertainty. The common philosophy is that if the object is not in its expected location, then a robot equipped with some sort of contact sensing should account for gaining additional information about the true object pose from contacts occurred when failed to grasp. Hence the majority of these algorithms poses the grasping problem as a stochastic decision-making problem in belief spaces (ISMDP). Belief planning requires selecting later actions taking into account the information gained by earlier actions and vice-versa. These methods are inefficient in practice - we will discuss the belief-planning paradigm in more details in Chap. 4. Typical implementations rely on simplifying the grasping problem by constraining the belief space to Gaussian distributions and construct sets with few actions and observations. As above mentioned, Gaussian distributions do not well-represent the uncertainty for real problems. Furthermore, extensions to non-parametrisation of belief spaces as in [Nikandrova et al., 2013] typically results in intractable planning problems due to the high-dimensionality of non-Gaussian parametrisation. In contrast, we built our approaches on [Platt et al., 2001], which allows us to construct on-line dexterous grasping trajectories more robust against pose uncertainty as well as to track high-dimensional belief states (see Chapter 6 for further details).

Section 2.6 discussed compliant control approaches to exploit contacts during the execution of reach to grasp trajectories. A class of approaches is based on mechanical

compliance derived by the hand design, i.e. use of elastic materials. These devices are cheaper, simple to control and are able to perform operations with no need to model uncertainty, but at the cost of lack of controllability. A second class of approaches implement a feedback control to mimic compliance. The drawback is the limits of the bandwidth of the controller, this means that the controller cannot react fast enough to external forces, thus no shock can be absorbed. For our grasp planner, we rely on a feedback controller (Chap. 7). This choice is forced by the fact that we need to know with accuracy the pose of our device in order to use contacts to refine our beliefs over the object pose.

In Sec. 2.8 we also presented studies on human grasping and manipulation. We showed that there is evidence that sensorimotor control in humans selects actions in order to maximise some reward function associated with the motor outcome. These models of optimal decision making provide an explanation on how humans compensate for uncertainty.

Finally, Sec. 2.9 discussed the problem of planning push operations, in which the planner has to compute motions for a robot but the desired motions belong to the object to be pushed. The relation between the space of object's motions and robot's control space (see Sec. 3.2.3) is typically intractable, but it is possible to build FMs to predict the motions that will result from a push operation. The push planner presented in Chap. 5 uses FMs to choose push operations to move 3D objects to a desired goal configuration.

Table 2.2: Grasp under uncertainty problem at glance

Problem	Methods	Pros	Cons	Sec
State Estimate	ML estimate	- Global estimator - Robust data correction	- No representation of uncertainty - Model-based	2.3
	Belief as PDF	- Representation of uncertainty	- Hard to estimate true distribution	2.3
	PF	- Non-parametrisation of belief	- Computational expensive	2.3
Grasp Synthesis	Analytical	- Pose as optimisation problem	- Computational expensive	2.4
	Learn by demonstration	- Require a teacher - Learn task operational space	- Large parameter space - Hard to generalise	2.4.2
	Object-centric learning	- Learn from experimental data - Generalise within and across object categories	- Learn from incomplete and erroneous data	2.4.2
Grasp Planning	Global policies	- Reason on informational effects - Posteriori analysis	- Belief space grows exponentially - Posteriori analysis is expensive	2.5.1
	Local policies	- Reason on informational effects - No posteriori analysis - Computational treatable	- Rely on ML observations - Require re-planning strategies	2.5.2
Grasp Execution	Mechanical	- Robust on collisions - Simple control - Do not need to model uncertainty	- Not adjustable compliance - Under-actuation - Reduced achievable postures	2.6.1
	Feedback control	- Adjustable compliance - Fully actuation	- Limited control bandwidth	2.6.2

Chapter 3

Path planning in deterministic domains

Motion planning refers to the problem of finding an appropriate path or trajectory to safely move an agent from a given starting pose to a desired final pose without collisions with obstacles. The agent in broad terms could be anything, from a mobile robot to a molecule. For the purpose of this thesis we will assume in our examples, though without loss of generality, that our robot is an open kinematic chain referred to as a manipulator (Sec. 3.2.6). More precisely, it is a dexterous manipulator composed of a humanoid arm and an anthropomorphic hand. A pose of the robot is considered to be in the *workspace* (Sec. 3.2.5) or *joint space* (Sec. 3.2.3) which defines both location and orientation in the workspace or a full description of the joint angles of the manipulator.

Complementary to the motion planning problem there is the challenge of determining a sequence of motor commands to follow a trajectory which moves the manipulator from its initial configuration to a desired one. This is referred as the *motion control* problem.

3.1 Organisation

The chapter proceeds as follows. First (in Sec. 3.2), we introduce the basic concepts and terminology from classical kinematics, such as degree of freedom, manipulator joint space, work space, and forward kinematics. Section 3.3 then addresses the problem of planning trajectories for robot manipulators, and we present techniques to avoid collisions with other bodies during path planning.

3.2 Robot design

A robot manipulator is composed by a series of rigid segments (or *links*) interconnected by *joints*. We assume a set of serially connected links[1] which form an open-chain manipulator. Joints usually consist of actuators and encoders. An actuator is the motor of the joint which enables movement of the next link in the chain, while an encoder keeps track of the current position of the joint. An actuated joint[2] is a controllable or *active* joint, otherwise a joint is an uncontrollable or *passive* joint.

3.2.1 Degree of freedoms of a manipulator

From classical mechanics the concept of *degree of freedom* (DoF) describes the number of independent variables used to uniquely determine the configuration of the system. A rigid body requires at least 6 variables to define its *pose* or *configuration* in a 3D Euclidean space[3]. We use the same concept in robotics to describe the number of independent variables used to specify the configuration of the robotic manipulator. The

[1] Links are connected one after the other.

[2] With the presence of an actuator or motor.

[3] Three variables to describe its position in x, y, z coordinates (for the $x-, y-, z-$axis respectively) and three variables to describe the orientation in terms of pitch, roll and yaw.

number of DoFs of a manipulator is strictly dependent on its number of joints and their type.

A tool or *end-effector* is typically attached to the end of the manipulator to interact with the environment. For dexterous manipulation we use an anthropomorphic robot hand which is composed of a fixed structure (*wrist/palm*) to which are independently attached open-chain *fingers*. The wrist is attached to the end of the manipulator therefore the arm and hand system is still an open-chain robot.

3.2.2 Manipulator joints

Whereas several joint types are available in literature, the most commonly used are the following:

- A *revolute joint* is by far the most commonly used. It is a 1DoF kinematic pair which provides a rotation about a single-axis rotation (Fig. 3.1(a));
- A *prismatic joint* is again a 1DoF kinematic pair and provides a linear translation along a single axis (Fig. 3.1(b));
- A *cylindrical joint* is a 2DoF kinematic pair and provides single-axis translation as well as single-axis rotation (Fig. 3.1(c));
- A *spherical joint* is a 3DoF kinematic pair and provides arbitrary rotation around a fixed point (Fig. 3.1(d)).

Fig. 3.1 shows graphically these most common joint types.

3.2.3 Manipulator configuration space

Given the kinematics of a robot manipulator (Sec. 3.2.6), its pose or configuration is uniquely determined by a vector $q \in \mathbb{R}^n$, where n is the degree of freedom (or DoF,

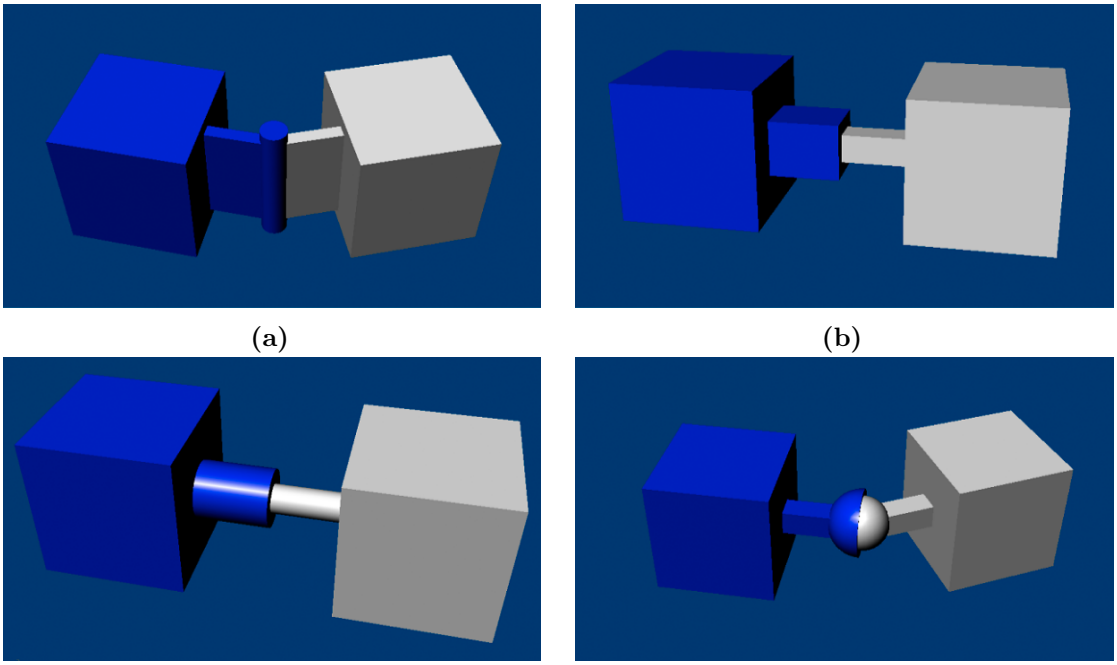


Figure 3.1: Common used joint types in robotics. Image (a) shows a 1DoF revolute joint which provides a single-axis rotation function. Image (b) shows a 1DoF prismatic joint which provides a linear translation function on a single axis. Image (c) shows a 2DoF cylindrical joint which provides single-axis translation as well as single-axis rotation. Finally, image (d) shows a 3DoF spherical joint which provides arbitrary rotation around a fixed point. Reproduced from [Kopicki, 2010]

Sec. 3.2.1) of the manipulator. In the rest of this thesis we will simplify the mathematical notation by assuming that joints with 2DoF or more can be represented by a combination of 1DoF joints[4] (Sec. 3.2.2). In explicit form, the vector q can be rewritten as

$$q = [\Theta_1, \Theta_2, \ldots, \Theta_n], \qquad \text{where } \Theta_i \in [a_i, b_i] \subset \mathbb{R}, \forall i \in [1, n] \in \mathbb{N}^+ \tag{3.1}$$

Joint variables, Θ_i, are subject to constraints on their interval of existence and a particular choice of a_i and b_i depends on the manipulator design. For example, revolute joints are naturally associated with a unit circle in the plane $\mathbb{S}^1$ such that $0 < b_i - a_i \leq 2\pi$.

[4] In this thesis we assume only revolute or prismatic joints depending on the case.

The set of all possible joint variables constitutes the space of legal configurations of the robot manipulator, which is often called its *configuration space*, or *joint space*, $\mathbf{C}$. The manipulator shown in Fig. 3.2 is composed of 6 revolute joints so that its joint space is a 6-torus $\mathbf{C} = \mathbb{T}^6$.

3.2.4 Manipulator workspace in joint space

An important aspect of the design of a robotic manipulator is the volume of space reachable by its end-effector. The *workspace*, $\mathbf{C}_{work}$, of a manipulator is the set of all the possible end-effector configurations that a manipulator can achieve in joint space. Note that the workspace $\mathbf{C}_{work} \subset \mathbf{C}$.

Note that a manipulator workspace can be split into two subspaces:

1. The *reachable workspace* is the set of possible end-effector configurations reachable by some choice of joint angles without considering the orientation of the end-effector;

2. The *dexterous workspace* is the set of possible end-effector configurations reachable by some choice of joint angle with arbitrary orientation of the end-effector.

From this definition is obvious that manipulators with less than 6DoF have an incomplete dexterous workspace, in terms of the map from joint space to end-effector poses, while manipulators with more than 6DoF have a degenerate one.

3.2.5 Operational workspace in SE(3)

The *operational workspace*, $\mathbf{W}$, sometimes referred to as the *world*, is the physical space in which the manipulator operates. Typical choices for the operational space are the Cartesian coordinate system in a 2D world (plane) or a 3D world. These choices are

usually sufficient for the majority of problems, but one might need to define a more complex world such as the surface of a sphere. For clarity, in this thesis $\mathbf{W} \subset SE(3)$ describes the pose of the end-effector in the environment. The reader should not confuse the operational workspace $\mathbf{W}$ with the workspace of the manipulator, $\mathbf{C}_{work}$, which is a vector of joint angles and identifies a region of the joint space reachable by the end-effector, as described in Sec. 3.2.4. The mapping between joint space and Cartesian space is given by the forward and inverse kinematics of the robot (Sec. 3.2.6). For brevity, I will use the term workspace throughout the entire thesis instead of operational workspace when there is no ambiguity.

3.2.6 Robot Kinematics

Robot kinematics is the study of the motion of a robot in terms of purely geometric constraints (i.e. leaving aside considerations of the relationship between forces, inertias and accelerations). In a kinematic analysis, the position, velocity, and acceleration of points on an open chain of links are computed without considering the forces that cause the motion. Instead, the positions and velocities of the end effector are related to angles and angular velocities at the robot's joints. Robot kinematics also deals with aspects of redundancy, collision avoidance, and singularity avoidance.

There are two types of robotic kinematic patterns, namely: (1) *forward* or *direct kinematics* and (2) *inverse kinematics*. The former determines the position of the robot's end effector with respect to a global frame of reference, given the length of each link and the angle of each joint. The latter, on the other hand, computes a set of joint angles to deliver the robot's end effector to a desired global position expressed in workspace, $\mathbf{W}$, given the length of each link.

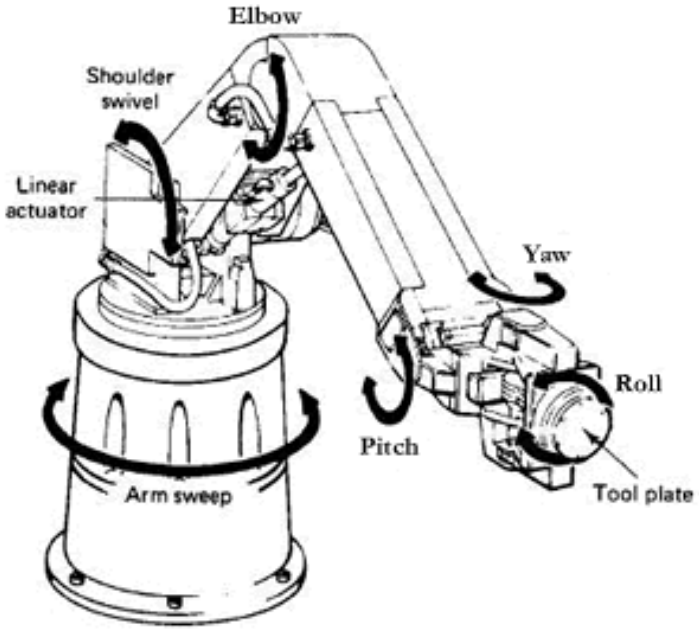


Figure 3.2: 6-DOF jointed arm robot. The basic three revolute joints enable arm swap, shoulder swivel and elbow rotations. In addition, three revolute joints allow the robot to point in many directions. [Web: ⟨http://eramandeepbansal.blogspot.co.uk/2013/08/mechanical-engg.html⟩]

3.3 Motion planning for robotic manipulators

The motion planning problem for a robotic manipulator refers to the problem of computing a trajectory to transfer the robot from an initial configuration to a goal configuration. Initial and goal configurations can either be specified in the joint space Sec. 3.2.3 or in the workspace Sec. 3.2.5 of a manipulator. Motion planning also requires us to determine the appropriate motor commands to follow a feasible trajectory.

Path planning is an instance of the motion planning problem in which differential constraints[5] are not considered. In the literature [see LaValle, 2006] this problem is often referred to as the *piano mover's problem* in which a piano needs to be moved from one room to another in a house, while avoiding collisions with the furniture and walls of the

[5]Constraints related to the velocity and the acceleration of a manipulator.

house. It is important to note that if there were no obstacles, linear interpolation would solve the problem efficiently.

The path planning problem for a robotic manipulator is formally introduced in Sec. 3.3.1. Among all the algorithms available for path planning the most common used are sampling-based approaches which perform approximate planning in continuous spaces using a finite set of samples. These sampling approaches can also help alleviate the problem of planning in high-dimensional spaces, which are subject to the *curse of dimensionality*.

3.3.1 Path planning

Let the *configuration space* $\mathbf{C}$, or $\mathbf{C}$-space, be a Euclidean n-space, a set of all possible reachable configurations of the robotic manipulator. The vector $q \in \mathbf{C} \subset \mathbb{R}^n$ identifies a unique configuration or pose (Sec. 3.2.3).

Let $\mathbf{W}$ be the *operational workspace* in which the manipulator operates, a Euclidean m-space. In a 2D world, then $m = 2$ (plane), and for a 3D world, $m = 6$ (Sec. 3.2.5).

The workspace $\mathbf{W}$ typically contains two different entities:

- The *obstacle region* is the subset $\mathbf{W_{obs}} \subset \mathbf{W}$ with the presence of "permanent" obstacles in the workspace;
- The *robot body* $\mathcal{A} \subset \mathbf{W}$ is a subset of the operational workspace occupied by the robot manipulator.

Both sets $\mathbf{W_{obs}}$ and $\mathcal{A}$ are typically represented via simple shapes, such as spheres and cubes, embedded in $\mathbb{R}^m$, where m is the dimensionality of the chosen *world*. Whereas $\mathbf{W_{obs}}$ is considered to be static, the planning problem requires the robot $\mathcal{A}$ to move, thus we will write $\mathcal{A}(q)$ as a function of its configuration q.

Since the planning happens in the configuration space $\mathbf{C}$ but the obstacle region $\mathbf{W_{obs}}$ is defined in the operational workspace $\mathbf{W}$, a mapping between these two spaces is necessary. The counterpart of the obstacle region $\mathbf{W_{obs}}$ in the $\mathbf{C}$-space is defined by

$$\mathbf{C}_{obs} = \{q \in \mathbf{C} : \mathcal{A}(q) \bigcap \mathbf{W_{obs}} \neq \varnothing\} \tag{3.2}$$

We are now ready to define the free space $\mathbf{C}_{free}$, in which we want the robot A to move avoiding collisions, as the complementary set of $\mathbf{C}_{obs}$ in $\mathbf{C}$,

$$\mathbf{C}_{free} = \mathbf{C} \backslash \mathbf{C}_{obs} \tag{3.3}$$

We define the path planning problem following the nomenclature in [Siciliano et al., 2008], given:

- The workspace $\mathbf{W} = \mathbb{R}^m$ with $m = 2$ or $m = 6$
- The obstacle region $\mathbf{W_{obs}} \subset \mathbf{W}$
- The robot body $\mathcal{A}(q)$ as a function of any configuration $q \in \mathbf{C}$
- An initial configuration $q_I \in \mathbf{C}_{free}$
- A goal configuration $q_G \in \mathbf{C}_{free}$

find a continuous path $\tau : [0, 1] \rightarrow \mathbf{C}_{free}$ subject to constraints $\tau(0) = q_I$ and $\tau(1) = q_G$.

Unlike planning problems over discrete configuration spaces, in which the configuration space is finite or countably infinite, path planning for robotic manipulation is naturally defined in continuous state spaces which means that the cardinality of $\mathbf{C}$ is uncountably infinite.

Whereas the general path planning problem over continuous spaces is not known to be

decidable yet, there are special cases in which is known to be solvable [Cheng et al., 2007]. For example, the simplified path planning problem known as the piano mover's problem, in which the obstacles and the robot are modelled as polyhedral, has been proved to be PSPACE-complete [Reif, 1979].

3.3.2 Sampling-based path planning

Path planning for a robotic manipulator requires us to search the configuration space in order to find a collision-free trajectory from the initial configuration to the goal configuration. From Sec. 3.3.1 we know that the configuration space is a Euclidean n-space where n is the number of DoF of the robotic manipulator. A manipulator for dexterous grasping typically consists of a robotic arm with not less than 6 DoF and a multi-fingered hand, which usually consists of 3 to 5 fingers with at least 3 DoF per finger. Thus the planning problem for dexterous grasping in robotics typically requires us to search a continuous, high-dimensional configuration space of 15 DoF even for the simplest manipulator.

A common practice is to use sampling-based approaches. The general philosophy is to avoid a complete representation of $\mathbf{C}_{obs}$ and conduct a search that probes the $\mathbf{C}$-space with a sampling scheme [LaValle, 2006]. This probing makes use of collision detection algorithms which are treated as a "black box" by the planner itself. Modern algorithms for collision detection enable us to compute efficiently whether a particular configuration $q \in \mathbf{C}$ is in free space. The planner, therefore, does not need any representation of the configuration space but delegates to the collision detection module the "responsibility" to either accept or reject samples.

Whereas sampling-based planners have been shown to be capable of solving real-world problems, they are not *complete*, which means that there is no guarantee that a sampling-based approach will discover a solution in a finite amount time even if there is one [Si-

ciliano et al., 2008; LaValle, 2006]. Instead, a weaker definition of *completeness* becomes critical: the concept of *denseness*, which is the ability to converge to a particular configuration as the number of sampling steps grows to infinity. Typically, sampling-based methods use a random sampling scheme which, by definition, is *dense* with probability one. A sampling-based algorithm is said to be *probabilistic complete* if the probability that a solution is found converges to one as the number of random sampling iterations goes to infinity. The reader should note that the probability completeness guarantees that a solution is found only if it exists, otherwise the algorithm may run forever.

Once a set of sample poses is defined, planning based on these is typically posed as a graph search problem. Let $\mathbf{G}(\mathbf{V}, \mathbf{E})$ be an undirected graph where $\mathbf{V} \subset \mathbf{C}_{free}$ represents the set of vertices, which are sampled configurations in free space, and $\mathbf{E}$ is the set of edges, which are collision-free paths. According to the way the graph $\mathbf{G}(\mathbf{V}, \mathbf{E})$ is constructed, we can distinguish two different approaches:

1. Single-query planners, such as RRT, which iteratively construct a *tree-like* data structure or a graph $\mathbf{G}(\mathbf{V}, \mathbf{E})$ from the initial configuration. Any new query requires building of a new data structure, but the exploration of the configuration space $\mathbf{C}$ is minimised.

2. Multi-query planners, such as PRM, which construct the graph $\mathbf{G}(\mathbf{V}, \mathbf{E})$ at the initialisation stage by assuming static obstacles. The graph is used as a fixed roadmap for multiple queries to minimise the computational time of each query.

3.3.2.1 Single-query planners

Single-query planners build a model of the robot $\mathcal{A}$ and the obstacle region $\mathbf{W}_{obs}$ at each query. However the exploration of the $\mathbf{C}$-space is limited until a solution is found. The fundamental idea is to construct a *tree-like* data structure rooted on a particular vertex

$q \in \mathbf{C}_{free}$. At each iteration, the exploration phase grows the tree in the reachable $\mathbf{C}_{free}$ space by adding a new leaf node q_{new} and a local path to connect q_{new} to the existing data structure. The procedure terminates when q_{new} is recognised as a solution.

The basic algorithm for single-query planners can be summarised by the following steps [LaValle, 2006]:

1. Initialise $\mathbf{G}(\mathbf{V}, \mathbf{E})$ by inserting at least one vertex in the set of vertices $\mathbf{V}$. Usually q_I, q_G or both are inserted. Other configurations in $\mathbf{C}_{free}$ may be included. The set of edges $\mathbf{E}$ is empty.

2. Choose a vertex $q_{curr} \in \mathbf{V}$ for expansion.

3. Select a configuration $q_{new} \in \mathbf{C}_{free}$, it may not matter if q_{new} is already a vertex of $\mathbf{G}$, and attempt to construct a local path $\tau_{local} : [0, 1] \rightarrow \mathbf{C}_{free}$ such that $\tau_{local}(0) = q_{curr}$ and $\tau_{local}(1) = q_{new}$.

4. If such a local path τ_{local} is found insert it into $\mathbf{G}(\mathbf{V}, \mathbf{E})$ as an edge, otherwise return to step 2.

5. Terminate the procedure if the graph $\mathbf{G}(\mathbf{V}, \mathbf{E})$ encodes a solution, otherwise return to step 2.

The type of tree constructed by the above procedure depends on the number of chosen root vertices:

- A *unidirectional* tree is rooted on a single vertex, usually the initial configuration q_I.

- A *bidirectional* tree requires two vertices as root, the typical use is to construct two trees rooted respectively on q_I and q_G.

- A *multidirectional* tree is rooted on more than two vertices, e.g. q_I, q_G and key configurations near narrow passages. Multidirectional trees provide several parallel

search procedures on different regions of the **C**-space, however in order to find a global path these trees need to be connected to each other, which may not be trivial in complex spaces.

A critical step in the above procedure is how to select a vertex from **V** for a further expansion of the tree.

La Valle proposed a class of algorithms known as *rapidly-exploring random trees* (RRTs) which can be intuitively considered as a Monte-Carlo search biased toward the largest Voronoi regions [LaValle, 1998]. RRT is a data structure and algorithm specifically designed to perform efficient searching in non-convex high-dimensional spaces. In robotics, RRTs are commonly used for path planning problems which involve obstacles and differential constraints such as non-holonomic robots or kinodynamics. RRTs are considered as planning techniques for generating open-loop trajectories for non-linear systems with state constraints.

RRTs have been extensively used and modified to improve their search behaviour in order to increase their performance on different problems.

Heuristically-guided random trees implement RRTs which are biased towards low-cost paths by using a heuristic cost function that computes the cost from the initial vertex and an estimate of the cost to go until the goal configuration [Urmson & Simmons, 2003]. Similarly Brock and his colleagues proposed *utility-guided random trees* that evaluate different aspects of the tree expansion procedure such as e.g. the utility value of the node to be expanded, the expansion distance, and the direction of the connection attempted [Burns & Brock, 2007].

3.3.2.2 Multi-query planners

The *Probabilistic RoadMap* (PRM) methods are the best known multi-query approach. They have been developed independently by different authors [Kavraki & Svestka, 1996; Geraerts & Overmars, 2002], but the general philosophy remains the same. The goal is to construct a roadmap in order to densely cover the free space $\mathbf{C}_{free}$. If the obstacle region $\mathbf{W}_{obs}$ is static, it is convenient to invest substantial computation time to generate the roadmap in order to efficiently answer new queries. The probabilistic roadmap planner consists of two phases: (1) a *construction phase* and (2) a *query phase*.

The construction phase can be summarised into the following steps [LaValle, 2006]:

1. Initialise $\mathbf{G}(\mathbf{V}, \mathbf{E})$ with empty sets of vertices $\mathbf{V}$ and edges $\mathbf{E}$.
2. Sample a configuration $q \in \mathbf{C}$ using sampling methods (Sec. 3.3.2.3).
3. If there are no collisions with $\mathbf{C}_{obs}$ insert q into $\mathbf{G}(\mathbf{V}, \mathbf{E})$ as a vertex.
4. Find the neighbouring set $\hat{\mathbf{V}}$ of q in $\mathbf{G}(\mathbf{V}, \mathbf{E})$.
5. For each vertex $q_i \in \hat{\mathbf{V}}$ attempt to construct a local path $\tau_{local} : [0, 1] \rightarrow \mathbf{C}_{free}$ such that $\tau_{local}(0) = q$ and $\tau_{local}(1) = q_i$.
6. If such a local path τ_{local} is found insert it into $\mathbf{G}(\mathbf{V}, \mathbf{E})$ as an edge.
7. Terminate the procedure if the number of vertices $\mathbf{V}$ has reached a predefined threshold, otherwise return to step 2.

The query phase can be summarised in two steps:

1. Add the initial and goal configurations from the query: $q_I, q_G \in \mathbf{C}_{free}$ to the graph $\mathbf{G}(\mathbf{V}, \mathbf{E})$.
2. Find the shortest path from q_I to q_G in $\mathbf{G}(\mathbf{V}, \mathbf{E})$ using an admissible graph search algorithm, e.g. Dijkstra's or A*.

3.3.2.3 Sampling techniques

In motion planning over continuous state spaces, the configuration space $\mathbf{C}$ is uncountably infinite, yet a sampling-based algorithm can use only countable samples. Therefore the way of drawing samples from $\mathbf{C}$ becomes critical in order to have a countable finite set of samples which adequately approximates the state space.

The most popular techniques are [LaValle, 2006]:

- *Random* sampling which draws points according to a uniform distribution.
- *Grid* sampling which spits the state space in equally sized cells. Each cell is then recursively split into smaller cells until the desired grid resolution is obtained.
- *Cell-based* sampling which attempts to uniformly spread samples in the workspace. Once a sample is generated, it is used to split the workspace in 2^m cells, where m is the dimension of the workspace (e.g. $m = 3$ for 3D spaces) and then the procedure is reiterated in each sub-cell.

In complex state spaces the obstacle region $\mathbf{C}_{obs}$ may be difficult to approximate with sampling techniques. Therefore several methods have been proposed to draw samples between obstacles or close to obstacle boundaries:

- *Voronoi region-based* methods use the notion of Voronoi regions for obstacles [Wilmarth et al., 1999; LaValle, 2001].
- *Gaussian sampling* methods generate points near obstacles using Gaussian distributions such that one point lies on $\mathbf{C}_{free}$ and the other on $\mathbf{C}_{obs}$ [Boor et al., 1999].

3.3.2.4 Collision detection

Sampling-based approaches rely on the ability to reject points that are in collision. Although the collision detection routine is typically treated as a black box, this constitutes the "bottle neck" of this kind of planning algorithm. Collision detection plays a critical role, especially when a local path to connect two nodes is computed. In practice, it is not possible to check all the possible configurations on a local path, however a fast and efficient collision detection method would be able to test a large number of discrete configurations along the path for better accuracy.

Modern collision detection algorithms avoid construction of a complete model of $\mathbf{C}_{obs}$ in order to reduce computational complexity. Testing collisions between simple geometric primitives, such as planes, spheres or cubes, is substantially faster than between non-convex polyhedra. Nonetheless, it may not be possible to represent a complex environment, up to an arbitrary degree of closeness with such simple shapes. A common approach is to model obstacles by using hierarchical structures of bounding primitives. Typically an obstacle is thus represented by a tree-like data structure where each node of the tree contains some sub-part of the geometry of the obstacle. These approaches can efficiently avoid unnecessary collision detection between bodies that are far apart [LaValle, 2006].

3.4 Conclusion

In this chapter we considered the problem of path planning in deterministic but continuous domains. Path planning requires us to search the configuration space for a collision-free trajectory that moves a robot from an initial configuration to a desired one. The search space is the joint space, or the space of all the possible configurations of the robot. This space is thus typically large. In this chapter, we discussed efficient

algorithms that find such trajectories by using sampling-based techniques, such as PRMs and RRTs.

In the next chapter, we will discuss how to formalise the problem of planning in stochastic environments as a decision-theoretic planning problem. The benefit of using this formulation is that decision theory provides a unified framework for many classes of planning problems. Nevertheless, in contrast with the techniques presented in this chapter, decision theoretic algorithms do not scale to some problems (continuous or partially observable environments) in a trivial way. We will thus discuss the potential of combining the path planning algorithms reviewed here, with the decision theoretic framework presented in the next chapter, so as to extend the latter to problems in continuous spaces.

Chapter 4

Planning under uncertainty

In robotics, action and perception are both affected by uncertainty, in terms of its state or the effects of actions on state. A robot's sensing abilities are imperfect and an exact model of the environment is not available. Hence an autonomous robot must act in *partial ignorance* and develop strategies that are robust with respect the uncertain effects that result.

There are several sources of uncertainty that can affect the problem of planning, however this thesis addresses only two:

1. *Uncertain in physical effects*: occur when the robot interacts with its environment via physical actions (e.g. contact operations). This interaction transforms the current state of the world accordingly to physical laws which may not be fully predictable. We can think of this as uncertainty in *future states*.

2. *Uncertain in informational effects*: occur when some of the quantities that define the *current state* of the world are not directly accessible to the robot. Thus the necessity to develop strategies to allow the robot to complete tasks despite partial ignorance by recovering knowledge of the state of the world. We can think of this

as uncertainty in the description of the current state of the system. Dealing with such an uncertainty is critical to develop (sub-)optimal strategies.

Chapter 3 reviewed *motion planning* algorithms that enable us to construct trajectories in continuous state spaces. These algorithms are concerned with the problem of finding collision-free trajectories to move a robot from an initial configuration to a desired one. In artificial intelligence (AI), planning originally meant a search for a sequence of logical operations or actions that transform an initial world state to a desired goal state [LaValle, 2006]. In this chapter, the notion of planning is extended beyond this to include many decision-theoretic ideas, such as *Markov decision processes*, to account for uncertainty. The basic idea is that our *decision maker* (DM) must be prepared to act in any state it may encounter, rather than only for the states along a single trajectory or path.

Decision-theory provides a unifying formulation for many classes of planning problems in AI. However, the computational complexity of these problems is such that it is treatable only for small size problems[1] [Papadimitriou & Tsitsiklis, 1987]. Robotic manipulation and grasping are typically defined over high-dimensional, continuous state and action spaces, hence the main concern in this thesis is to investigate the space of possible solutions that combine the best feature of motion planning and decision-theoretic approaches. The former enables us to efficiently search in high-dimensional, continuous states whilst the latter provides the theoretical framework to construct robust strategies under uncertainty.

4.1 Organisation

The flow of this chapter proceeds as follows. Section 4.2 introduces the concept of an embedded system in which a DM interacts with its environment via actions. Actions

[1]The size of the problems refers to the cardinality of the state, action and observation spaces.

make the environment transition from one state to another, and it is assumed that the environment is static, in the sense that no transition is allowed unless triggered by an agent's action. This is the basic model used throughout this thesis.

Section 4.3 formulates a decision-theoretic framework that is appropriate for the problem of planning under uncertainty about physical effects. In this framework, our main concern is to determine some notions of *feedback*. Since the effects of an action are not fully predictable, a DM requires sufficient sensing capabilities to infer these effects *a posteriori*.

Section 4.4 extends such a framework to the case in which a robot's sensing abilities are no longer sufficient to recover precisely the current state of the world. Thus an alternative formulation is presented which moves the problem from the familiar state space to a richer space named *information* or *belief space*. Although the belief space is continuous even for underlying discrete state spaces, the benefit of planning in beliefs is that the agent is now capable of reasoning about the informational effects of its own actions. Nevertheless these methods are exact but inefficient[Papadimitriou & Tsitsiklis, 1987].

Section 4.5 presents hybrid methods used to approximate solutions for decision-theoretical problems in belief space. This section mainly focusses on how to construct suboptimal local strategies under the assumption of deterministic dynamics of the system. This means that the DM builds its solutions assuming that the most likely observation will always occur. Typically these techniques provide some sort of failure recovering strategy, such as re-planning, to be used when this assumption turns to be not true.

Section 4.6 summarises the conclusions of this chapter and describes the contributions the frameworks presented here make to the subsequent chapters.

4.2 A robot and its environment

In this thesis, a robotic system is composed by a robot and its environment (see also [Littman, 1996; Wyatt, 1997]). These two entities are considered as an embedded system, rather than two separated systems. The robot takes as input the state of the world - or, in the case of imperfect sensing, some noisy observations of the state - and generates as output actions which themselves affect the state of the world. It is important to note that in embedded systems the future sequence of states of the environment is influenced by the agent's outputs at earlier stages.

Throughout this thesis the world is considered to be static, in the sense that no change of state is possible unless the robot performs an action.

4.3 Planning for physical effects

Let us consider the idea of uncertainty on physical effects as a "game against nature" [LaValle, 2006]. In such a formulation, whilst our robot makes decisions about which action to perform, another DM called "nature" makes decisions to determine the values of all the uncertain parameters, such mass, friction *et ect.* The evolution of the system is then fully determined according to both of these decisions. Our concern is to build a model to describes how nature makes its decisions, so that we can construct robust strategies with respect to it. In the literature, this is called the *uncertainty model.*

Uncertainty models are typically classified as follows:

- *Non-deterministic*: a set of possible outcomes is provided. Since any of the outcomes are equally possible, this model usually leads us to build "pessimist" strategies which always expect the worst case outcome. This model is adequate when

interacting with a strong antagonist [LaValle, 2006].

- *Probabilistic*: uncertainty is described by a conditional probability distribution over possible outcomes, given certain conditions. These models are well-suited for uncertainty due to noisy sensors.

- *Deterministic dynamics*: either assumes that the most likely outcome always occurs, or a mapping between state-action pairs and outcomes is known, typically this mapping is learnt. These models are typically used to build approximate solutions and thus necessitate recovery strategies.

The approaches presented in this thesis are consistent with either probabilistic or deterministic dynamics. In the next section a variety of stochastic decision-theoretic classes will be presented in more detail.

4.3.1 Stochastic discrete-state Markov Decision Processes

In general terms, a discrete stochastic process is defined by a set of random variables $\{X_t, t \in \mathbf{T}\}$, where $\mathbf{T} = \{1, 2, \dots\}$ is the set of possible times and X_t denotes the outcome at the t^{th} stage or time step. The domain of X_t is the set of all the possible outcomes denotes $\mathbf{S} = \{s_1, s_2, \dots, s_N\}$, where $N = |\mathbf{S}|$ is the cardinality of the set $\mathbf{S}$.

A *Markov Decision Processes* (MDP) [Littman, 1996; Kaelbling et al., 1998] models such class of problems in which the agent has complete and perfect perceptual abilities. A MDP is described as a tuple $\langle \mathbf{S}, \mathbf{A}, \mathcal{T}, \mathcal{R} \rangle$, where

- $\mathbf{S}$ is a finite set of states of the world;

- $\mathbf{A}$ is a finite set of actions;

- $\mathcal{T}$:$\mathbf{S} \times \mathbf{A} \rightarrow \Pi(\mathbf{S})$ is the state transition function, giving for each world state and agent action, a probability distribution over the world's states. Therefore

$\mathcal{T}(s, a, s')$ is the probability that performing action a in state s will lead to state s', $\forall s, s' \in \mathbf{S}$, and $\forall a \in \mathbf{A}$;

- $\mathcal{R}: \mathbf{S} \times \mathbf{A} \to \mathbb{R}$ is the reward function, giving the expected immediate reward gained by the agent for taking each action in each state. Equivalently, we may define a *loss* or *cost* function $\mathcal{J}$ that we attempt to minimise.

In accordance with the Markov property, this model assumes that the next state and the expected reward depend only on the current state and the action taken, hence

$$Pr(s_{t+1}|a_t, s_t, a_{t-1}, s_{t-1}, \ldots, a_1, s_1) = Pr(s_{t+1}|a_t, s_t), \quad \forall s_i \in \mathbf{S}, a_i \in \mathbf{A}, i \in [1, t+1]$$

A solution for a decision-making problem is defined by determining an action-selection strategy for an agent in all the possible situations it may encounter. The reward function $\mathcal{R}$ encodes measures of desirability for performing actions in particular states, and thus the behaviour of the DM is affected by the choice.

4.3.2 Constructing a reward function

The behaviour of a decision theoretic model is strongly related to our choice of the reward function, $\mathcal{R}$, which must be selected by hand for each problem. Defining $\mathcal{R}$ can be difficult and it is more an art than a science. *Utility theory* studies the problem of preferences and defines in a formal way the existence of, and how to construct (if possible) a reward or cost function for rational DMs. See [Barbera et al., 1999] for further details on this subject.

In general terms, an expected immediate reward function, $\mathcal{R}(s, a)$, encodes the relative desirability of performing an action a in state s. Or, in other words, it is the desirability of ending in state s' by performing action a in state s. Thus $\mathcal{R}(s, a)$ can be thought of

as a normalised weighted sum of the outcomes, as follows:

$$\mathcal{R}(s,a) = \sum_{s' \in \mathbf{S}} \mathcal{T}(s,a,s')\mathcal{R}(s',a)$$

where $\mathcal{R}(s',a)$ is a function that expresses in numerical values the desirability of being in state s' after performing the action a.

As an example, let us consider the case in which our aim is to reach a goal region $\mathbf{S}_G \subset \mathbf{S}$, no matter which action leads us to the solution (i.e. uniform cost for each action). In this simple example we can define our desirability or reward function as

$$\mathcal{R}(s') = \begin{cases} +1 & \text{if } s' \in \mathbf{S}_G \\ 0 & \text{if } s' \notin \mathbf{S}_G \end{cases}$$

The next question is how to build action-selection strategies that are optimal with respect to a particular reward function. Sec. 4.3.3 answers to this question.

4.3.3 Acting optimally in a MDP

An optimal solution for an MDP is obtained by choosing a policy $\pi \in \mathbf{\Pi}$ that maximises the expected cumulative rewards, typically the expected discounted sum over a lifetime or *horizon*.

If the DM has a limited lifetime, these types of solutions are referred to as fixed *finite-horizon* and typically the policy, δ, is defined as a time-dependent or *non-stationary* policy. Such non-stationary policies are usually composed of a set of *stationary* policies, $\delta = \langle \pi_T, \ldots, \pi_1 \rangle$, where the set of indices $[T, \ldots, 1]$ refers to the *horizon length* or number of steps left. Intuitively, the way the agent selects its final decision is different from the way decisions are made when there are many steps left.

On the other hand, in the *infinite-horizon* formulation an agent does not know when it will cease to gain rewards, therefore there is no need to change its policy as a function of time.

For simplicity, let us consider the infinite horizon formulation. Although the optimal policy π^* is what we need to solve a MDP, most MDP solvers derive the policy from a *value function*, V. The value function encodes the utility of following a particular policy π. In the case of infinite-horizon the value function is defined as a discounted cumulative reward, as follows

$$V^{\pi}(s) = \mathcal{R}(s, \pi(s)) + \gamma \sum_{s' \in \mathbf{S}} \mathcal{T}(s, \pi(s), s') V^{\pi}(s') \tag{4.1}$$

where $\gamma \in (0, 1]$ is a discount coefficient.

This means that the optimal policy, π^* can be written in terms of its value function V^{π^*} (written as V^* in order to lighten the mathematical notation) as

$$\pi^*(s) = \arg\max_{a \in \mathbf{A}} [\mathcal{R}(s, a) + \gamma \sum_{s' \in \mathbf{S}} \mathcal{T}(s, a, s') V^*(s')]$$

Similarly, we define the optimal value function by a set of equations

$$V^*(s) = \max_{a \in \mathbf{A}} [\mathcal{R}(s, a) + \gamma \sum_{s' \in \mathbf{S}} \mathcal{T}(s, a, s') V^*(s')] \tag{4.2}$$

The set of equations defined by 4.2, also known as *Bellman equations*, are not linear, due to the presence of the maximisation operator, hence it is not sufficient to use simple techniques like Gaussian elimination to solve them. The next section will introduce a classification of the most common algorithm used to solve stochastic discrete-state MDP problems.

4.3.4 Solving discrete MDPs

Although many algorithms have been developed for solving discrete MDPs, this section will only report a few basic algorithms: *value iteration* (VI), *policy iteration* (PI) and *Q-learning*. A more complete survey is available in [Littman, 1996; Puterman, 1994; Wyatt, 1997]. In Sec. 4.5.1, we will discuss some possible ways to extend this framework to continuous state and action spaces.

4.3.4.1 Value iteration

As the name suggests, this algorithm is based on the intuition of deriving an optimal policy directly from the value function as described in Eq. 4.2. Typically the algorithm proceeds by recursively computing the sequence of discounted finite-horizon value functions, V_t, until the maximum difference between two consecutive value functions, the *Bellman error magnitude* or *Bellman residual*. are less than a user-defined threshold.

Typically such methods are implemented in a backward fashion. The first iteration computes the expected immediate reward for the final time step of the problem for each state-action pair. No further computation is required since there are no future actions to be taken. The second iteration deals with the problem of computing a value function of horizon length of 2, which is the expected immediate reward plus the value for the next chosen action. Conveniently, the value for the next action has already been computed during the first iteration. The algorithm holds to this procedure until the termination condition is met. In general, each iteration takes $|\mathbf{A}||\mathbf{S}|^2$ steps [Littman, 1996]. The number of steps executed depends on the problem but it has been proved to be polynomial in $|\mathbf{S}|$ [Littman, 1996].

4.3.4.2 Policy iteration

Instead of deriving a policy from the value function as in the value iteration approach (Sec. 4.3.4.1), the PI algorithm works directly in the space of policies. At each iteration, a full policy is generated. The intuition is that the algorithm is composed by two steps: *policy evaluation* and *policy improvement*. The former evaluates the current policy, π, by using the utility function V^{π}, whilst the latter uses this information to construct a better policy that strictly improves the utility function associated with it. The algorithm terminates when no improvement can be achieved.

Each policy evaluation step can be performed in $O(|\mathbf{S}|^3)$, while the policy improvement step takes $O(|\mathbf{A}||\mathbf{S}|^2)$. Since there are $|\mathbf{A}|^{|\mathbf{S}|}$ policies, policy iteration requires in the worst case $|\mathbf{A}|^{|\mathbf{S}|}$ iterations, however in the majority of cases, it is proved to converge no more slowly than the VI algorithm [Littman, 1996; Puterman, 1994].

4.3.4.3 Q-learning

The algorithms presented in the previous sections (Sections 4.3.4.1 and 4.3.4.2) show how to find an optimal policy an MDP given a complete description of its states, actions, rewards and critically the transition function. On the other hand, the Q-learning algorithm and its variants are *temporal difference* algorithms for finding optimal policies by learning from experience. The intuition is that an agent is capable of learning how to act by interacting with its environment.

Q-learning can be thought as a sampling-based method for estimating the optimal state-action values, or Q function, for an unknown MDP. The Q function is defined as

$$Q^*(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \epsilon \mathbf{S}} \mathcal{T}(s, \pi_t(s), s') V^*(s') \tag{4.3}$$

where $V^*(s')$ is defined as in 4.2. The agent uses the experience collected by acting in the environment to improve its estimate $\hat{Q}$. The experience represents a single action and can be summarised in a tuple $\langle s, a, r, s' \rangle$: the agent starts in state s, performs action a, receives reward r and ends in state s'. Then $\hat{Q}$ is improved according to the Q-learning rule

$$\hat{Q}(s,a) = (1-\alpha)\hat{Q}(s,a) + \alpha(r + \gamma \max_{a' \in \mathbf{A}} \hat{Q}(s',a'))$$

with the *learning rate* $0 < \alpha \leq 1$.

4.3.5 Remarks on MDPs for robot pushing

The theoretical formulation presented in Sec. 4.3.1 well-suits the problem of planning physical interactions with an environment, such as pushing operations (see Chap. 6). The state transition function $\mathcal{T}$ describes the uncertainty model over the set of parameters chosen by nature, such as mass and friction of the object to be pushed, friction of the supporting surface, *et etc.* Since the true values of this set of parameters are typically unknown, it is hard to determine how an object will behave under a push operation. Nevertheless, this uncertainty model can either be learnt (i.e. [Kopicki et al., 2011]) or a physics simulator can be used as a black box (i.e. [Zito et al., 2012b]) (see also Chap. 6 for further details). Either way, it is important that the DM can fully determine the current state of the system at any time in order to recover for unexpected outcomes.

In practice, this formulation has several limitations. First, the assumption of perfect sensing abilities cannot be applied to real problems; as an example, in the scenario of planning pushing operations, a real robot would need to track the pushed object by noisy sensors (such as cameras). This limitation is easy to overcome by extending the model to partially observable environment as described in Sec. 4.4.1. A more important limitation is the computational cost of computing a solution. In an intuitively way,

finding an action-selection strategy that acts in any possible situations an agent may be, means that all possible situations have been already visited and all the possible outcomes have been adequately evaluated. The MDPs suffer from the *curse of dimensionality*, in the sense that the effort of finding a solution depends on the number of states and actions, and on the intrinsic complexity of the problem itself. Problems in continuous space are intractable by using this formulation. However it is possible to construct approximated solutions as discussed in Sec. 4.5.1.

4.4 Planning for informational effects

The MDP formulation presented in Sec. 4.3.1 addresses the problem of choosing optimal actions in *completely observable stochastic domains* [Toussaint et al., 2011], in robotics we need a model more suited to observation uncertainty. For example, in the grasping problem, once the robot has perceived the object to grasp (e.g. using vision) there is typically residual uncertainty about geometric and physical properties of the object, such as pose, shape, mass, friction. Thus the robot should define and plan a sequence of actions to reach a target grasp configuration and close the fingers in such a way to maximise the likelihood of a successful grasp. To do this, the robot needs a formal model of sequential decision making which describes how to gather (task-related) information, if necessary, and how the belief states evolve once this new information is acquired.

In practice, we are considering the problem of choosing optimal actions in *partially observable stochastic domains*. Problems like the one described above can be modelled as *Partially Observable Markov Decision Processes* (POMDPs) [Littman, 1996; Kaelbling et al., 1998]. In a POMDP, the system interacts with a stochastic environment whose state is only partially observable. Actions change the state of the environment and lead to numerical penalties/rewards, which may be observed with an unknown temporal

delay. Similarly to an MDP, the model aims to devise a policy for action selection that maximises the reward - or minimises a cost function.

4.4.1 Stochastic discrete-state Partially Observable Markov Decision Processes

Formally, a POMDP can be formalised as a tuple $\langle \mathbf{S}, \mathbf{A}, \mathcal{T}, \mathcal{R}, \mathbf{O}, \mathcal{Y} \rangle$, where $\mathbf{S}, \mathbf{A}, \mathcal{T}$ and $\mathcal{R}$ keep the same definition mentioned above. In addition:

- $\mathbf{O}$ is a finite set of observations;
- $\mathcal{Y}: \mathbf{S} \times \mathbf{A} \rightarrow \Pi(\mathbf{O})$ is the observation function, giving for each world state and agent action, a probability distribution over the set of observations. Therefore $\mathcal{Y}(s, a, o)$ is the probability that performing action a in state s will lead to observation o, $\forall s \in \mathbf{S}$, and $\forall a \in \mathbf{A}$, and $\forall o \in \mathbf{O}$;

Obviously, the POMDP framework embraces a large range of practical problems. However, solving a POMDP is intractable: finite-horizon POMDPs are PSPACE-complete [Papadimitriou & Tsitsiklis, 1987] and infinite-horizon POMDPs are undecidable [Madani et al., 1999].

4.4.2 Acting optimally in a POMDP

An important feature of a POMDP is the assumption that the agent has no direct access to the states of the environment. It has to infer its knowledge from some sequence of observations that give incomplete information about the current state. In accordance with [Ross et al., 2008], a complete history of the system at time t is defined as:

$$h_t = \{a_1, o_2, \ldots, o_{t-1}, a_{t-1}, o_t\} \tag{4.4}$$

This explicit representation of the past is typically memory expensive. It is possible to

summarise all the relevant information in a probability distribution over the state space **S**. In the literature, the probability distribution over states (as well as over actions or observations) is referred as a *belief state*, b, and the entire probability space (the set of all possible probability distributions over the set of physical states) as the *belief space*, $\mathcal{B}$. A belief state at time t is defined as the posterior distribution over the physical states, given the complete history and some prior belief:

$$b_t(x) = \Pr(x_t = s \mid h_t, x_1) \quad \forall s \in \mathbf{S} \tag{4.5}$$

The belief b_t is a sufficient statistic for the history h_t [Smallwood & Sondik, 1973]. Therefore, the agent is able to choose the current action according to its current belief state and the initial belief, b_1. In the AI literature, POMDPs in belief spaces are known with several names, such as *Information State MDPs* (ISMDPs) or belief-state MDPs. I will refer to them as ISMDPs. At any time t, the belief state b_t can be computed, by using *Bayesian filtering*, from the previous belief state b_{t-1}, the previous action u_{t-1} and the current observation y_t, as follows:

$$b_t(x_t = s') = \Pr(y_t|x_t = s') \sum_{s \in \mathbf{S}} \Pr(x_t = s'|a_{t-1}, x_{t-1} = s) b_{t-1}(x_{t-1} = s) \tag{4.6}$$

Once a way to compute the current agent's belief state is defined, the important question is how to use this information for choosing an action at any time t. This action is determined by the *policy* π which specifies the probability of using each action in any given belief state. Given a belief filter there is a deterministic optimal belief policy. An optimal strategy maximises the expected sum of discounted rewards until the time horizon T, as follows:

$$\pi^*_T = \arg\max_{\pi \in \Pi} E[\sum_{t=1}^{T} \gamma^t \sum_{x \in X} b_t(x) \sum_{a \in \mathbf{A}} \mathcal{R}(x, a) \pi(b_t, a) | b_1] \tag{4.7}$$

Where $\gamma \in [0,1)$ is the discount factor and $\pi(b_t, a)$ is the probability that action a is performed in belief b_t according to the policy π. In a similar way, we also compute the utility function, or Bellman equation (see Sec. 4.3.3), obtained following the policy π:

$$V^{\pi}(b) = \sum_{a \in \mathbf{A}} \pi(b,a)[\mathcal{R}(b,a) + \gamma \sum_{o \in O} \Pr(o|b,a) V^{*}(\mathcal{T}(b,a,o))] \tag{4.8}$$

where now $\mathcal{R}(b,a)$ is the immediate expected reward over belief states and actions, and $\mathcal{T}(b,a,y)$ is the belief state transition function, giving for each belief state, action and observation, a probability distribution π over belief states. The optimal policy π^{*} defined in equation 4.7 to be the strategy that maximises the equation 4.8, formally one can write this as:

$$V^{*}(b) = \max_{a \in \mathbf{A}} [\mathcal{R}(b,a) + \gamma \sum_{o \in \mathbf{O}} \Pr(o|b,a) V^{*}(\mathcal{T}(b,a,o))] \tag{4.9}$$

Another useful quantity is the belief-action $Q(b,a)$-value which defines the value of a by assuming that the optimal policy is followed at every step afterwards.

$$Q^{*}(b,a) = \mathcal{R}(b,a) + \gamma \sum_{o \in \mathbf{O}} \Pr(o|b,a) V^{*}(\mathcal{T}(b,a,o)) \tag{4.10}$$

The introduction of imperfect sensing abilities adds new complications in this framework. However, the formulation based on informational states is a neat, elegant way to pose the problem of acting in partially observable environments. If the belief space is a sufficient statistics, the Markov property is guaranteed to be true, and the ISMDP formulation can be thought of as an MDP in a continuous state space (where now the states are belief states). The next section gives an overview of the most common approaches used to solve ISMDPs.

4.4.3 Remarks on POMDPs for robot grasping

The benefit of planning with beliefs is the ability to reason about the informational effects of sequences of actions. In typical belief space planning this means performing a kind of pre-posterior analysis, in which planned actions cause imagined observations that are in turn used to perform a Bayes' update of the belief state. As the reachable belief space grows exponentially in the length of the planned action-observation sequence, these methods are exact but inefficient [Papadimitriou & Tsitsiklis, 1987].

Let us consider again the problem of robotic grasping introduced earlier. As we have seen, the robot is equipped with noisy sensors, therefore it is necessary to develop strategies that can cope with the residual uncertainty on physical and geometrical properties of the object to be grasped. Let us assume that a complete model of the object is available. This means that only the pose of the object remains uncertain. Nevertheless, this simplified problem presents several complications that prevent us from finding a solution for the associated ISMDP problem. One of the complications is the dimensionality of the action space. In fact, the action space is defined as the space of possible trajectories in the joint space of the manipulator, and for such large spaces also sampling techniques fail to produce a dense approximation of the original space. This problem will be discussed in more details in Sec. 4.5.1.

4.5 Hybrid models

This section describes hybrid models in which approximate POMDP approaches are applied to high-dimensional continuous spaces or path planners for high-dimensional spaces are extended to cope with state uncertainty.

4.5.1 Stochastic decision process in continuous spaces

This section introduces a set of algorithms for finding a solution for large ISMDPs. By large, I mean that either the state space, the action set, or both are continuous. The set of observations can be assumed to be a countable finite set without losing information. For example, in robotic grasping, the set of observations may represent the tactile contacts for each finger of the robot hand.

In section 4.3.4, we discussed a set of algorithms for finding optimal solutions for a MDP assuming a set of finite states and actions. However, solving a stochastic decision problem is intractable: finite-horizon ISMDPs are PSPACE-complete [Papadimitriou & Tsitsiklis, 1987] and infinite-horizon ISMDPs are undecidable [Madani et al., 1999], hence extensions to the continuous state and action formulation is not trivial. The main problem is that finding an optimal solution requires us to iterate over the elements of the state set, $\mathbf{S}$, and action set, $\mathbf{A}$. The key intuition behind most solutions to the extended problem is to find a finite approximate representation of the state and action space in order to successfully apply similar methods to the ones described in Sec. 4.3.4. Typically, a finite representation is obtained via a sampling-based technique, such as the ones described in Sec. 3.3.2. A broader survey of approximate algorithms for solving large POMDP problems is available in [Ross et al., 2008].

An efficient way to break the curse of dimensionality for these problems is to bias a sampling method to regions of interest of the belief space. Instead of searching for an approximate solution for all possible situations an agent may encounter, another option is to address the problem of solving a specific instance of the problem. In other words, given a current initial belief state, which characterises a particular instance of the problem, all the computational efforts are used to explore only the region of belief space that is possible to reach from the initial state. The notion of *reachability* is defined via the state transition function. In the literature these methods are also called *on-line*

methods, in contrast to the *off-line* methods which attempt to construct solutions for any possible situation. Tables 4.1 show an overview of the principal off-line and on-line methods.

It is important to notice that the concept of *reachability* can be extended to the related action space. In fact, for some problems, it is possible to define a finite set of actions applicable in a given state. As an example, let us consider the problem of planning pushing operations. In this case it is possible to define a set of linear trajectories for the robot's finger for which a push is generated [Zito et al., 2012b]. Although there are infinite trajectories, a sampling-based technique can span the action space in order to generate a push for any possible directions. Yet, for the problem of grasping, a set of actions can be pre-computed as in [Hsiao & Kaelbling, 2010], however this means that for different object shapes a different set of actions has to be computed. In contrast, this thesis addresses the problem of build a trajectory on-the-fly which is more robust in the face of uncertainty (this approach is discussed in Chapters 6 and 7).

The SARSOP algorithm and its variants extend the concept of reachability to the notion of an *optimally reachable space*, $\mathbf{R}^*(b_1)$, that is the region of belief space reachable by following an optimal policy [Pineau et al., 2006]. Since $\mathbf{R}^*(b_1)$ is not known in advance, SARSOP iteratively refines its approximation, $\hat{\mathbf{R}}^*(b_1)$ by using a variation of the VI algorithm for large state spaces, named point-based value iteration (PBVI) algorithm. Initially the SARSOP algorithm relies on heuristic exploration of the reachable belief space, $\mathbf{R}(b_1)$, and over time improves its sampling strategy by learning which part of the belief space is worth exploring. Additional to its heuristic exploration, the efficiency of the algorithm can be improved by providing approximations of the upper and lower bounds for the utility function, computed by using off-line methods. The proof of concept is applied in a variety of robotic tasks as well as grasping (e.g. [Hsiao et al., 2007]). See Sect. 2.5.1 for a more detail discussion on the application of these methods to the problem

of planning robotic grasping.

Another variation of PBVI algorithm is presented in [Spaan & Vlassis, 2005]. The authors proposed PERSEUS, a randomised PBVI algorithm for POMDPs. The basic concept of the PBVI algorithm is to approximate the belief space through sampling. Once a finite set of belief states is generated, the PBVI algorithm proceeds as the VI algorithm discussed in Sec. 4.3.4.1. In contrast, PERSEUS only computes the approximated value function on a (randomly selected) sub-sampled set of belief states, ensuring that at each iteration the value function is improved (or at least is not decreased) for all the belief states. The same idea can be used to extend this approach to problems with a continuous or large action space. When the action set is finite and small, one can run an optimisation procedure to maximise the value function according to the set of transitions, observations and rewards. PERSEUS adopts a sampled-max operator that performs the maximisation over a randomly sampled set of actions. However, this means that PERSEUS requires a parametrised model that computes the set of transitions, observations and rewards given any selected action. This approach has been demonstrated for navigation problems with continuous state and action spaces [Spaan & Vlassis, 2005].

The work shown in [Thrun, 2000] proposed another approximate approach, the *Monte Carlo POMDP* (MC-POMDP) algorithm, which can accommodate real-valued spaces and models. The author is interested in POMDPs with continuous state and action spaces in order to generalise the model for a large number of real-world problems that are continuous in nature. The central idea is to use Monte Carlo sampling for belief representation and propagation, while reinforcement learning in belief space is employed to learn value functions, using a sample-based version of nearest neighbour for generalisation. Empirical results illustrate that this approach finds close-to-optimal solutions efficiently. Furthermore, initial experimental results demonstrated that this approach is

applicable to real-valued domains and that it yields good performance results in environments that are, by POMDP standards, relatively large.

In [Silver & Veness, 2010] is presented the state-of-the-art for online planning in large POMDPs, termed *Partially Observable Monte-Carlo Planning* (POMCP). The algorithm uses sampling techniques to break the curse of dimensionality by combining a Monte-Carlo update of the agent's belief with a Monte-Carlo search tree (MCST) from the current belief state. Moreover, the work shows that only a black box simulator of the POMDP model is required rather than an explicit probability distribution over states, actions and observations. Unlike previous methods, this technique provides scalable performance to a variety of challenging problems such as 10×10 battleship and partially observable PacMan with approximately 10^{18} and 10^{56} states respectively. POMCP consists of a *Upper Confidence Bound applied to Tree (UCT)* search [Kocsis & Szepesvari, 2006] that selects actions at each time-step and a particle filter that updates the agent's belief. In practice, each state of the MCST is viewed as a multi-armed bandit and actions are chosen by using the UCB1 algorithm [Kocsis & Szepesvari, 2006]. It is important to note that the UCT algorithm has been extended to partially observable environments by using a search tree of histories instead of states. However, the POMCP algorithm is based on a *rollout policy* to update the agent's belief. In other words, in order to evaluate the benefits of selecting a particular action a in a particular state s, observing a one-step evolution of the system is not sufficient. A rollout policy is a heuristic-based policy that is used to explore possible future evolutions of the system and use this information to better evaluate our initial choice a.

4.5.2 Path planning under uncertainty

Path planning, as introduced in Sec. 3.3.1, addresses the problem of finding a collision-free trajectory to move a robot from a given starting pose to a goal pose. Chapter 3

showed how to plan trajectories in continuous, high-dimensional state spaces assuming deterministic dynamics of the environment and perfect sensing abilities of the robot. This section extends the concept of path planning algorithms to cope with various kinds of uncertainty in the search space. Note that the set of vertices, V, and edges, E, used in path planning algorithms can be thought as the counterpart for the set of states, $\mathbf{S}$, and actions, $\mathbf{A}$, respectively, in the discrete state decision-theoretic formulations considered earlier in this chapter.

Most of the proposed algorithms for path planning under uncertainty explicitly consider uncertainty in the domain, in a similar way to a particle filter. In this kind of approach a belief state is substituted by a set of particles or hypotheses, P, in which each particle determines a possible configuration of a robot. An observation model is needed to predict how state uncertainty effects action execution and how the belief state evolves. The majority of these algorithms have been applied to the problem of *navigation* for mobile robots in uncertain terrains, e.g. [Melchior & Simmons, 2007].

As an example, the *Particle RRT* (pRRT) uses a particle filter representation of the state space (robot's poses in this case) to encode uncertainty in a traditional RRT algorithm (discussed in Sec. 3.3.2.1). In this work, the authors addressed the problem of navigation planning assuming that the physical properties of the terrain are uncertain. The goal is to find paths more robust in the face of uncertainty in order to maximise the likelihood of a successful execution. The basic principle of the algorithm is to plan paths which are inherently safer because a robot can follow them more closely despite unknown characteristics of the terrain. This algorithm has been demonstrated in simulation for a rover operating autonomously in rough terrains with unknown coefficients of friction [Melchior & Simmons, 2007].

The key idea is similar to the formulation adopted in this thesis for planning under uncertainty in physical effects (Sec. 4.3): an external DM called nature makes decisions

about the coefficients of friction, given a particular action (i.e. motor command) for the robot and nature's choice, the outcome of the action would be deterministic. Since the choice of nature is unknown, a probabilistic model over possible friction coefficients is required. The search tree (RRT) grows exploring the uncertain environment in order to find a safe path to move the robot to a desired location. Each RRT extension, or local path to connect two RRT nodes, is treated as a stochastic process, thus the same action, from the same robot's starting pose, is simulated several times with different values for the coefficient of friction. Hence areas of the map associated with narrow probability distributions - i.e. surfaces with well characterised friction - are expected to produce similar behaviours over many simulated trials.

Prentice and Roy have also investigated the problem of planning safer paths under imperfect sensing information [Prentice & Roy, 2008]. As in SARSOP, the key intuition is to efficiently compute the reachable part of belief space. However, in general, tracking the belief evolution can be computationally expensive, and the notion of reachable belief regions depends on the initial belief state. The authors proposed an efficient method to overcome both these problems. They showed that a PRM can be constructed for a graph representation of the reachable belief space from an initial query, and then re-used for future queries. However, this formulation works only for linear Gaussian state estimation problems. In fact, the probability distribution over states is assumed to be Gaussian and that the transition and observation functions are assumed to be linear with Gaussian noise, so that the belief filter can be implemented as the Kalman filter. Note that for simple non-linear transition and observation functions is possible to extend this formulation by using an extended Kalman filter (EKF). In addition, the factorisation of the covariance matrix of the EKF yields a linear update stage in the belief representation, hence it is possible to predict in a single step statistical information such as the mean and covariance of the resulting belief state after a sequence of actions and observations. This single prediction step is used for efficiently planning motions between two nodes of

the PRM.

4.6 Conclusion

This chapter presented a formulation for the problem of planning robotic grasping and manipulation under uncertainty. Two kinds of uncertainty have been discussed, namely: uncertainty in physical effects and uncertainty in informational effects. The former occurs when the robot interacts with its environment and some physical parameters are unknown. In this case it is hard to predict the evolution of the system. The latter occurs when the robot has no direct access to the state of its environment, therefore it is necessary to reason on how to gather information in order to improve its knowledge of the environment and complete a task.

Markov decision processes are well-developed frameworks for modelling action-selection strategies under these types of uncertainties. However, the computation complexity of these problems is such that they are intractable in practice. On the other hand, path planning algorithms provide efficient methods to explore large spaces. The main aim of this thesis is to combine the benefits of path planning algorithms and the elegance of decision processes in uncertain environments, in order to develop more robust planning strategies in the face of (prediction and sensing) uncertainty for robotic tasks such as grasping and manipulation.

Chapter 5 addresses the problem of planning push operations for a robotic manipulator equipped with a single finger. The goal is to find a sequence of pushes to move a target object from an initial configuration to a desired one. This problem is hard to solve because it is difficult to predict how an object to be pushed will behave under a particular push operation. The sets of states and actions are defined on continuous spaces, hence formulating the problem as a stochastic decision problem is not trivial.

In contrast, this chapter presents my approach to this problem: a hierarchical planner composed of two levels. The high-level planner is based on a single-query path planning algorithm (here an RRT) that efficiently explores the state space of the object to be pushed, whilst a low-level planner deals with the problem of selecting push operations to achieve a desired motion of the object.

Chapters 6 and 7 address the problem of planning reach to grasp trajectories for a dexterous robotic manipulator under sensing uncertainty. This problem cannot be directly formulated as an ISMDP because of the high-dimensionality of state and action spaces. Chapter 6 presents a novel approach to plan grasping trajectories whilst simultaneously maximising the likelihood of gathering information on the location of the object to be grasped. The algorithm constructs a PRM in the joint space of the robot manipulator and plans trajectories assuming that the most likely observation will always occur. At execution time, if an unexpected observation occurred (here tactile contacts), then a new trajectory is re-planned. Chapter 7 presents an extension of this algorithm to also cope with shape uncertainty and generate smoother re-planning trajectories.

Table 4.1: Principal on-line and off-line methods to solve POMDPs.

OFF-LINE METHOD	PROPERTIES
Point-Based Value Iteration (PBVI)	- Maintains only a set of belief states - Only considers constraints that maximise the VI for at least one example - Define a lower bound
MDP	- Does not consider the uncertainty of the states - Defines an upper bound
QMDP	- Defines a single linear piecewise convex (LPWC) approximation for each action - The uncertainty of the states disappears after a single step - Define a upper bound
Fast Informed Bound (FIB)	- Defines a single linear piecewise convex (LPWC) approximation for each action - Belief update considers only at some degree the partial observability of states - Defines a tighter upper bound than QMDP
ON-LINE METHOD	**PROPERTIES**
Branch-and-Bound Pruning	- Uses AND-OR tree - Maintains lower and upper values of $Q^*(b, u)$ for every belief and action in the tree - Back propagated
Monte-Carlo Sampling (MC-POMDP)	- Reduces the branch factor at only the belief state reached during the simulation - The simulator may be a black box
Heuristic Search	- Expands only fridge nodes in according with the heuristic

Chapter 5

Path planning for physical effects

In the previous chapters two different planning problems were investigated, namely: *path planning* and *planning under uncertainty*. As we discussed, path planning concerns the problem of constructing actions (typically trajectories in free space) for a robot in deterministic, but continuous domains. In contrast, planning under uncertainty concerns the problem of finding action-selection strategies that are more robust in the face of uncertainty, but generally these techniques are only adequate for discrete domains.

This chapter formalises the problem of planning a sequence of push operations (action-selection), by which a robotic arm equipped with a single rigid finger can move an object to be manipulated towards a desired goal pose. This is a challenging problem because of the complex relationship between pushing actions and resulting object motions. A plan must be built in the action space of the robot, which is only indirectly linked to the motion space of the object through a complex interaction for which inverse models may not be known.

I will present a two stage approach to planning pushing operations. A global RRT path planner is used to explore the space of possible object configurations, while a local push

planner makes use of predictive models of pushing interactions, so as to plan sequences of pushes to move the object from one RRT node to the next. The effectiveness of the algorithm is demonstrated by simulation experiments in which a robot must move a rigid body through complex 3D transformations solely by applying a sequence of single finger pushes.

5.1 Organisation

This chapter proceeds as follows. Section 5.2 provides an introduction to the problem of robotic pushing and addresses the problem of planning in domains where it is hard to derive inverse models. Additional information is required of what pushes should be applied to move the object along the sequence of planned poses. There is often no way of knowing what the outcome of a push will be, except by trialling it in a predictor (here for proof of principle we use a physics simulator (Nvidia Physx), however learned predictors may also be useful [Kopicki et al., 2011]). Instead, what is needed is a planner, which uses a physics engine to span the working space with a branching tree-like structure of possible object motions. Note that all predictive models are not exact, thus we need to somehow overcome such a source of uncertainty. In this chapter, the planner overcomes this source of uncertainty by re-planning: if the observed object pose deviates from the planned pose at execution time, then a new plan is constructed starting from the present pose.

Section 5.3 presents the core algorithm of this chapter. This algorithm is based on the RRT algorithm presented in Sec. 3.3.2.1. A tree-like structure is constructed in the configuration space of the object to be pushed, as in the standard algorithm. However, for this problem, each extension of the tree can be thought of as a MDP in continuous spaces under uncertainty on the physical effect of the action to be applied. To overcome

the problems of solving such a large MDP problem, I present an alternative formulation based on a randomised depth-first search procedure.

Section 5.4 presents experimental test results in which a series of pushes are applied to objects and both learned and physics-based predictors are tasked with predicting the resulting motions. Performance is evaluated through a combination of virtual experiments in a physics simulator.

5.2 Introduction

In robotics, as described in [LaValle, 2006], motion planning was originally concerned with problems such as how to move a piano from one room to another in a house without colliding with obstacles or bounding walls. The state-of-the-art algorithms to solve such problems are based on randomly sampling points or poses in the free configuration space of the object being moved, e.g [Kavraki & Svestka, 1996; Geraerts & Overmars, 2002].

While a path planning algorithm can deal with the large state space to find a sequence of intermediate poses of the object (waypoints) to move the object from an initial pose to a desired one, finding a sequence of pushing to move the object from one waypoint to another is not trivial. As mentioned in Sec. 4.3.5, this is a decision-making problem, however, the complexity for searching for a policy in continuous state and action space render such a formulation impractical. The next section explains how this problem can be transformed in an optimisation problem and solved efficiently.

5.3 Planning pushes to reach a goal pose

The method presented here breaks this process down into two components. First, a *global* path planner is considered, which uses an RRT algorithm to explore the configuration space of the object to be pushed by growing a tree of nodes towards the desired goal pose. Secondly, a *local* push planner is discussed, which uses a randomised depth-first search procedure for finding local sequences of pushes to reach from a previous node towards the next candidate node suggested by the RRT. Note that although I describe this work as a 'two-level planner', a third level of planning is also used in the sense that conventional motion planning techniques - here a state-of-the-art PRM - are applied to build any particular action that the arm will execute. However, since building simple motion actions for a conventional arm is well understood and deterministic, 'two-level' refers to the key problem of choosing which set of push actions to build.

5.3.1 Global path planner

RRT planners iteratively expand the search tree by applying control inputs which lead the system towards randomly selected points. The RRT planner considers these selected points as new *candidate* nodes and tries to extend the closest vertex (node) of its existing search tree towards them. The important point is that the RRT planner does not directly compute the actions that are required in order to extend its tree, but instead selects new candidate nodes (in our case object poses) which must be moved towards by applying an appropriate sequence of pushes.

As described earlier and in [LaValle, 1998], this kind of planning can generally be viewed as a search in a metric space, $\mathbf{X}$, for a continuous path from a given initial configuration, x_{init}, to a target configuration, x_{goal}. In conventional motion planning, a state transition function of a form $\dot{x} = f(x, u)$ is enough to encode the kinematic and dynamic constraints

Algorithm 1 BUILD_RRT

Input: $x_{init}, \Delta t, \mathbf{G}$
Output: T
  T.init(x_{init})
  repeat
    x_{rand} ←RANDOM_STATE()
    x_{near} ←NEAREST_NEIGHBOUR(x_{rand}, T)
    π ←LOCAL_PUSH_PLANNER($x_{near}, x_{rand}, \Delta t$)
    x_{new} ←NEW_STATE(x_{near}, π)
    T.add_vertex(x_{new})
    T.add_edge(x_{near}, x_{new}, π)
  until SATISFIED($x_{new}, \mathbf{G}$) or maximum number of nodes, ζ_1, reached

of the problem. The vector u is typically selected from a set of possible control inputs, $\mathbf{U}$. In the simplest case, u can be analytically computed and specifies a particular direction in the metric space in which we extend the RRT. The vector $\dot{x}$ represents the derivative of the state x with respect to time. A new state x_{new} which will become a new node of the search tree is simply computed by integration of $\dot{x}$ over a fixed time period Δt. Note that, in contrast to Chap. 4, I denote the continuous state and action spaces with, respectively, $\mathbf{X}$ and $\mathbf{U}$, instead of the more familiar notation $\mathbf{S}$ and $\mathbf{A}$ used for discrete spaces.

Since we are not able to explicitly compute the correct action to achieve a desired object motion, we make use of physics engines to predict the outcome of a possible action. Hence, an intuitive approach is to try some pushing actions in different directions until we find one which causes the object to move in a desirable direction. More specifically, in our scenario the space $\mathbf{X} \subseteq SE(3)$ represents the set of all possible configurations of the object with respect to a global frame of reference o. Any point $x \in \mathbf{X}$ is expressed as $x = [R_o, p_o^T]$, where $R_o \in SO(3)$ is a rotation matrix and $p_o \in \mathbb{R}^3$ is a translation vector, both over the x, y and z axis with respect to o. Hereafter we also refer to this set as the *configuration space*.

Unlike with more conventional uses of RRTs, we need to specify the control inputs of

Algorithm 2 LOCAL_PUSH_PLANNER

Input: $x_{near}, x_{rand}, \Delta t$
Output: π

```
x ← x_near
π ← ∅
repeat
  Û ←SAMPLE_RANDOM_ACTIONS(x, N)
  x_best, u_best, ρ_best ←SELECT_ACTION(x, Û, x_rand, Δt)
  if u_best = ∅ then
    store as failure and goto repeat
  end if
  π.add_element(x_best, u_best, ρ_best)
  x ← x_best
until ρ_best < ϵ or maximum iterations, ζ_2, reached or maximum number of failures, ζ_3, reached
```

the system in a different space that should not be confused with the configuration space of the object. Motion planning for the manipulator occurs in the *joint space*, $\mathcal{C}$, which defines the set of all possible configurations that the robot arm can assume. In particular, we want to select the change in the joint space, Δu, which produces a desirable change in the object configuration space, Δx, such that the reduction of the distance between the current object pose and the next candidate node, selected by the RRT algorithm, is maximised as follows:

$$\bar{u} = \arg\min_{u \in \hat{\mathbf{U}}} \rho(x_{rand} - f(x, u, \Delta t)) \tag{5.1}$$

where $\hat{\mathbf{U}}$ is a set of randomly selected actions (as described in the next section); x_{rand} is the next candidate node, randomly selected by the RRT algorithm for expansion of its tree; $f(\cdot)$ is the state transition function which applies the action u in the current state x for a fixed time interval Δt; and ρ is a metric distance in $\mathbf{X}$ which denotes the distance between two configuration states $x, x' \in \mathbf{X}$ in terms of rotational and translational displacement.

The rotational or angular displacement between two object poses is evaluated using the quaternion representation. Let q, q' be points on the 3D unit sphere about the origin in 4D quaternion space, which represent the rotation matrices $R, R' \in SO(3)$ respectively, then it is possible to evaluate the difference in orientation between the two poses as:

$$\|q - q'\|_{rot} = \frac{2}{\pi} min(\cos^{-1}(qq'), \cos^{-1}(q(-q'))) \tag{5.2}$$

The rotation space is isomorphic to the 3D unit sphere in quaternion space, with diametrically opposite points identified [Mason, 2001]. The distance is computed as the smaller of the two possible angles between the images of the two rotations on this sphere. The distance thus is never greater than $\frac{\pi}{2}$. For simplicity we normalise the value of the angular distance into the range $[0, 1]$.

We denote the translational distance between two different positions $p, p' \in \mathbb{R}^3$ as the Euclidian norm in $\mathbb{R}^3$:

$$\|p - p'\|_2 = \sqrt{(p_1 - p'_1)^2 + (p_2 - p'_2)^2 + (p_3 - p'_3)^2} \tag{5.3}$$

Now, given two object poses $x, x' \in \mathbf{X}$, we are able to evaluate the "distance" between them as a combination of the two aforementioned metrics as follows:

$$\rho(x, x') = \frac{1}{2}\|Q(x) - Q(x')\|_{rot} + \frac{1}{2L}\|p - p'\|_2 \tag{5.4}$$

where $p, p' \in \mathbb{R}^3$ are the translational components of x, x' respectively; $Q(x)$ is an operator which transforms the orientation of a pose x into the corresponding point on the surface of a 3D unit sphere in quaternion space; and L is a critical parameter of the workspace (e.g. the diameter of the region in which the robot manipulations take place). The purpose of L is to ensure that both rotational errors and translational errors

Algorithm 3 SELECT_ACTION

Input: $x, \hat{\mathbf{U}}, x_{rand}, \Delta t$
Output: $x_{best}, u_{best}, \rho_{best}$
  $\rho_{best} \leftarrow \infty$
  $x_{best}, u_{best} \leftarrow \varnothing$
  for all $u \in \mathbf{U}$ **do**
    $x' \leftarrow$ SIMULATE_ACTION$(x, u, x_{rand}, \Delta t)$
    $\rho' \leftarrow \rho(x_{rand}, x')$
    if $\rho' < \rho_{best}$ **then**
      $x_{best}, u_{best}, \rho_{best} \leftarrow x', u, \rho'$
    end if
  end for

Algorithm 4 SIMULATE_ACTION

Input: $x, u, x_{rand}, \Delta t$
Output: x'
  $x' \leftarrow x$
  repeat
    $x \leftarrow x'$
    $x' \leftarrow f(x, u, \Delta t)$
  until $\rho(x_{rand}, x') \leq \rho(x_{rand}, x)$

occupy ranges between 0 and 1, so that the summation of (5.2) results in a meaningful cost function, in which rotational and translational displacements result in costs of similar magnitude. Note that alternative methods of computing a net pose error are also possible, such as that adopted in [Kopicki, 2010].

Algorithm 1 shows the pseudo-code of our global path planner which iteratively build an RRT, T, given an initial pose of the object. The main difference between our implementation and the standard RRT planner described in [LaValle, 1998] consists of the choice of the control inputs. In our case, the variable π identifies a sequence of pushes instead of a single vector.

5.3.2 Local path planner

Algorithm 2 shows the implementation of a *local* push planner which estimates a sequence of motor commands which will move an object to the next candidate node x_{rand} (which has been randomly generated by the RRT), from a pose at the closest vertex x_{near} of the existing tree structure. The RRT planner supplies both inputs: the candidate node x_{rand} and the closest vertex x_{near} to it, determined according to the metric ρ described above.

The local push planner now randomly selects N possible finger trajectories in $\mathcal{J}$ which ensure that the end effector of the manipulator will collide with the object. Each trajectory N is generated as follows in the procedure SAMPLE_RANDOM_ACTIONS(x, N). First a pair of points are randomly generated, each from a uniform distribution on two different surfaces of the object, neither of which is in contact with any other surface. These are then linked along a straight line path, which is extended away from the object in space by a fixed distance d from each surface point. Thus a pair of points in configuration space have been defined that define the beginning and end of a straight line path through the object. These are then converted into joint space via an inverse kinematics solver, and a conventional PRM based path planner [Kopicki, 2010] with optimisation is used to generate the trajectory in joint space. All the N trajectories are simulated using a physics engine. Algorithms 3 and 4 define how the planner selects the next manipulative action by making use of a simulator for prediction. The physics simulator provides a prediction of the next resulting object configuration x', given the current configuration x (initialised as the vertex x_{near}) and the selected action u for a fixed time interval Δt, $x' \leftarrow f(x, u, \Delta t)$.

During the simulation, the actual object's configuration state is checked at each fixed time period Δt. After each period Δt, if the distance ρ to the candidate node x_{rand} has been reduced, then action u is continued for an additional period Δt. Once the distance

ρ begins to rise again, rather than continuing to fall, the push is interrupted and the configuration which minimised the distance is stored. Note that the pose at which the push is interrupted may often be an unstable pose in which the object is balanced along an edge while resting against the finger. For this reason we instead find the closest stable pose by withdrawing the finger and letting the object settle into a stable configuration. This stable configuration is the one that is stored as the new node on the RRT.

If the final stored configuration is identical to the initial configuration, x_{near}, then it means that this action is not useful and the planner discards it. The simulator then resets the initial position of the object at x_{near} and repeats the procedure by executing each of the remaining N pushing actions. After all N push trajectories have been simulated, the planner selects the most efficient push which minimises the distance ρ to the candidate node x_{rand}. If no trajectory reduces ρ, then the planner stores this iteration as a failure and selects other N random pushing actions. If a preset limited number of failures is reached without reducing ρ, then the RRT planner will not extend the tree, and will instead randomly choose a new candidate node x_{rand}.

Once a candidate node, x_{rand}, and a useful pushing action u have been found (where u moves the object nearer to x_{rand}, thus diminishing ρ), it is still unlikely that u alone will bring the object sufficiently close to x_{rand} (satisfying a pre-defined maximum distance). As suggested by the results of this study (described in Section 5.4), it is more likely that a single action will move the object to an intermediate configuration which is still close to the initial one, x_{near}. If we settled for the single push u alone, and this intermediate configuration was added to the RRT as a new node, we would lose the desirable behaviour of the RRT planner that yields a biased exploration towards unexplored regions. Applying a naive random action-selection behaviour to the RRT planner would transform its behaviour into a simple naive random tree, where randomly selected vertices are extended by random actions in order to generate new vertices. In this case, as

described in [LaValle, 1998], "Although one might expect the tree to randomly explore the space, there is actually a very strong bias toward places already explored". Consequently most of the new generated nodes would remain in the same Voronoi region as their parent nodes.

To avoid this overlapping or clustering of nodes, it is necessary that the local push planner be iterated to extend these 'intermediate' nodes until a configuration pose is found which comes close to the candidate pose, x_{rand}, which has been requested by the RRT planner (specified by some maximum threshold distance). The intermediate nodes are encoded as part of a control sequence which defines a *series* of pushes (in the experiments presented here, typically 2-3 pushes) to extend the vertex x_{near} towards the new node x_{rand}.

Once a series of pushes has been found that achieves a configuration x_{best} which comes suitably close to x_{rand}, the series of pushes is recorded and x_{best} is added as a new vertex to the RRT tree structure. This process continues until an RRT node is found that comes sufficiently close to the overall goal state of the manipulative operation. Figures 5.2(a)-5.2(b) compare our algorithm and a naive RRT which iteratively extends the tree using a single push within the $N = 8$ randomly generated options. The results clearly show that, although the naive RRT is iteratively less expensive in terms of computational time, it converges very slowly with respect to the solution computed by the approach presented in this chapter.

Note that using a physics simulator as a forward model for pushing can be problematic, since it will not perfectly predict the results of pushes on real objects. Furthermore, reasonable predictions require the careful tuning of a large number of physical properties (e.g. friction coefficients) which will not generalise to new objects or scenarios. Nevertheless, we employ a physics engine here for proof of principle, as a simple and convenient means of forward modelling. Other work, [Kopicki et al., 2011], suggests

that superior predictions (with superior generalisation capabilities) can be enabled by *learning* forward models, and we intend to substitute these prediction methods, instead of the physics engine, in our future work.

Additionally, note that *all* forward models (either physics-based, [Duff et al., 2011], learned, [Kopicki et al., 2011], or a combination of both, [Kopicki et al., 2010]) make predictions that can have significant errors. The planner overcomes this source of uncertainty by re-planning: if the observed object pose deviates from the planned pose for the present node by more than a predefined threshold value, then a new plan is constructed starting from the present pose.

5.4 Results

This section presents the empirical experiments and the results collected to evaluate the presented algorithm.

5.4.1 Experiments

The push planning algorithm was tested using a simulation environment based on the NVIDIA PhysX physics engine. The test environment features a simulation model of a five axis manipulator (modelled after the Neuronics Katana 320 robot, [Neuronics AG, 2004]), equipped with a single rigid finger with a spherical finger-tip, see Fig. 5.3. While PhysX and other physics simulators do not perfectly replicate real-world rigid body motions, [Kopicki et al., 2011], the simulated motions are physically plausible, and important physical properties are modeled, including static and dynamic friction coefficients (which for proof of principle we have hand-tuned in our system), collisions, and gravity.

In each experiment, the robot is tasked with pushing an L-shaped object (referred to as a "polyflap" [Sloman, 2006]) from a randomised starting pose, towards a randomly chosen goal pose. Polyflaps are interesting objects for testing push manipulation planners, because they can occupy a variety of different stable configurations, as well as a range of unstable modes. They are free to tip and topple, as well as slide and rotate upon a plane. In particular, we distinguish between the two flanges of each polyflap (shown as different colours in the figures), so that it is often necessary for the robot to 'flip' the polyflap in order to move it into the desired goal configuration. This choice of object enables us to illustrate the planned 3D manipulations, in contrast with the majority of related literature in which the manipulanda are effectively 2D and are constrained to planar motions of sliding and rotating upon a flat surface.

Small five-axis manipulators, such as the Neuronics Katana, have a very limited envelope of effective operation. By coding this constraint into the planning algorithm, we see that the system plans pushes, both towards and away from the robot, in such a way that the object is never pushed out of the effective reach of the arm.

5.4.2 Examples

Figure 5.3 shows an example of experiments in which a sequence of pushes has been planned which successfully manipulates a polyflap into a desired goal configuration. In each image sequence, the wire framed polyflap (towards the right of each image) is a 'phantom' which is merely used to denote the desired goal configuration. The two faces of the polyflap are distinguished, one in pink and the other in grey, so that the reader can understand the orientation of the object following each successive push by the robot.

A detailed explanation of the sequence of pushes and resulting object motions is provided in the caption under each figure. Note (by observing the pink and grey faces of the

polyflap) that the robot successfully re-orients the object by causing it to flip its resting base from one face to the other, in order to achieve the desired goal configuration.

5.4.3 Trends and evaluation data

For proof of principle 30 experiments were carried out in which the robot had to plan and execute a series of multiple pushes. In all experiments the robot successfully maneuvered the polyflap to within the specified threshold distance from the desired goal configuration.

During these experiments the global path planner described in Algorithm 1 was set with a maximum number of nodes $\zeta_1 = 2000$, and the method SATISFIED(x_{new}) returns *true* only for configurations within 6 cm and 9 degrees[1] of translational and rotational displacement (respectively) from the goal configuration. The local push planner described in Algorithm 2 was set with a threshold value $\epsilon = 0.01$, a maximum number of iterations $\zeta_2 = 3$ and a maximum number of failures $\zeta_3 = 3$. The only parameter varied in the experiments was the number N of randomly trialled pushing trajectories at each iteration (N is defined in section 5.3.2). Ten experiments were performed each for N equal to 4, 8 and 12 pushes respectively.

For each experiment data was collected on the total number of iterations, the total number of generated RRT nodes, the rotational and translational distances and the combined cost function values each time a new RRT node has been generated. This data is charted in Figures 5.1 and 5.2. Figures 5.2(a)-(a) show how the error between the achieved object pose and the goal pose decreases as successive nodes are added to the RRT tree. Figure 5.2(b) shows how the computational cost increases as additional nodes are added to the RRT tree. The convergent properties of the results suggest that an

[1] In the experimental results, the rotational error s computed on the unit hypersphere in 4D quaternion space, thus the displacement between two orientations is the length of the arc of the geodesic which connects them. In Fig. 5.1(b), 9 degrees are equivalent to 0.01.

arbitrary degree of positional accuracy is achievable, with an expected trade-off between desired end-state accuracy and computational burden.

The results in Figures 5.1 and 5.2 reveal trade-offs between the amount of effort expended in high-level planning (generating additional RRT nodes) versus effort expended in low-level planning (computing the pushes required to move between RRT nodes). Additionally, a naive implementation of the RRT planner is also shown, which only generates a single push ($\zeta_2 = 1$) to connect one RRT node to the next (in contrast, the other methods use multiple pushes, typically three, between RRT nodes). The naive RRT results (labelled nRRT, Figures 5.1 and 5.2) show that this approach damages the useful exploratory properties of RRT planners, since new nodes requested by the RRT planner are often not reachable with a single straight-line push. This causes the RRT to generate excessive numbers of nodes, which tend to be closely clustered rather than reaching out to explore the search space. Similar sub-optimal behaviour is also observed in the "RRT with 4 pushes". This is because, even though this method allows multiple inter-node pushes, the space of selectable pushing actions is very sparse, resulting in crude and inefficient plans. This explains why poorer accuracy and a large number of nodes result from both "RRT with 4 pushes" and the naive "nRRT with 8 pushes".

In contrast, RRT "with 12 pushes" explores 12 different possible push directions at each low-level iteration, with the benefit that comparatively few pushes are required to move from one RRT node to the next. From Figures 5.1 and 5.2 it appears that there is an optimal trade-off "sweet spot" between these extremes, with RRT "with 8 pushes" proving the most efficient.

As we expected the number of trialled pushing trajectories available at each iteration affects the accuracy and computational time of the local push planner. A planner with a larger set of actions is supposed to spend more time in exploring the outcome of all possible actions, whilst a more accurate exploration allows the planner to find the

goal configuration in fewer steps. In particular, the results show that the RRT with 4 pushes generates a poor exploration which means that the performance, in terms of cost reduction (Chars 5.2(a)- 5.1(a)) and computational time (Chart 5.2(b)), is typically worse than any other planner I tried. This suggests that it requires a larger threshold of maximum iterations, ζ_2, in order to move the polyflap from x_{near} to x_{rand} within the given threshold value ϵ. That obviously also affects the entire solution path, which requires more nodes and longer computational time to achieve the same accuracy as the other planners.

5.5 Conclusion

This chapter has addressed the issue of how to plan a series of actions of a robot manipulator to achieve a transformation for an object. Previous work was restricted to 2D motions [Cappelleri et al., 2006]. My approach, by contrast, splits the planning problem into two parts: a global RRT planner; and a local planner which performs a randomised depth-first search procedure in the action space of the robot. The push plans are iteratively refined by using a physics simulator to evaluate them. I have shown the ability of this two level approach to produce plans for a push manipulation scenario. Empirical experiments suggest convergence of the planned action sequences on arbitrarily accurate approximations to the desired goal state.

As a remark, two aspects of the algorithm presented should be discussed: i) the set of parameters and ii) the choice of the cost function. First, I present the set of parameters divided in two categories:

Global planner parameters:

- ζ_1: maximum number of nodes in the tree;
- problem constrains.

Local planner parameters:

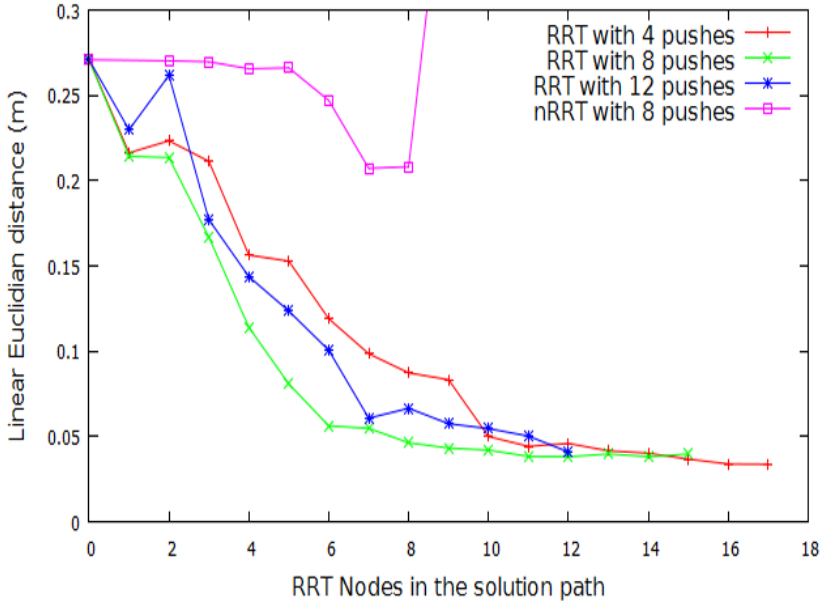
- ζ_2: maximum number of iterations for the local planner;
- ζ_3: maximum number of failures for the local planner;
- ϵ: tolerance for the local push planner;
- N: number of pushes tested at each iteration;
- L: normalising factor for the cost function;

The former set of parameters are typical terminal conditions for an RRT-like planner, to avoid that the algorithm run forever. The first terminal condition is not of particular interest for the scope of this discussion, however a special note should be made for the second one. In many applications an analytic solution of the problem at hand is not available, however a set of (sufficient) constraints can be defined such that: i) any given state that satisfies these constraints is a solution, and ii) this procedure can be performed in polynomial time w.r.t. the number of bits used to described a state. In this thesis, I define such constraints as translational and rotational tolerance w.r.t. a given goal state. This choice has also another advantage: "entire path planning algorithms can be constructed without requiring the ability to steer the system between two prescribed states, which greatly broadens the applicability of RRTs" [LaValle, 1998].

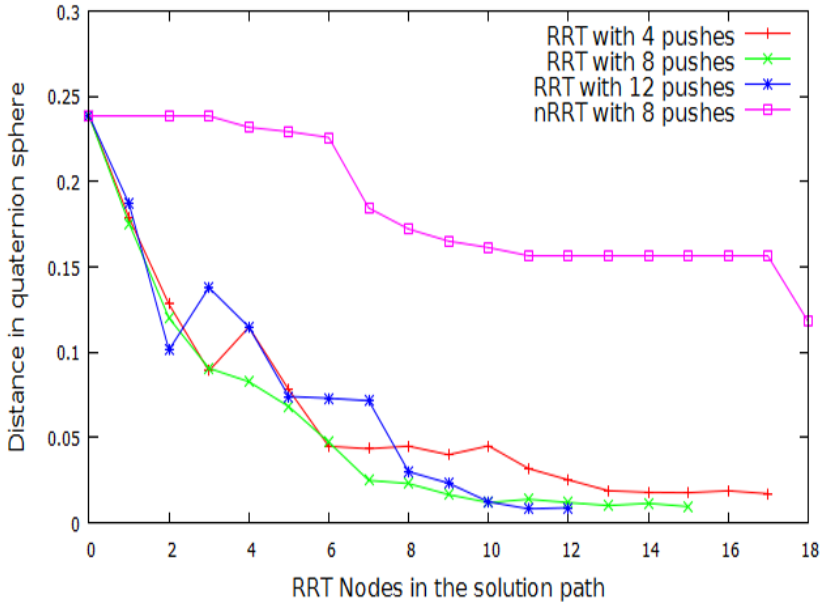
On the other hand, the set of parameters for the local planner is critical for the behaviour of the proposed algorithm, especially the maximum number of failures (ζ_3) and the size of the action set used to extend the tree (N). In Sec. 5.4, I have empirically shown how accuracy and computational expense vary with different choices for such parameters. As already mentioned, ζ_3 affects the ability of the local push planner to reach a candidate node x_{new}. When $\zeta_3 = 1$ the exploration is biased towards already visited regions of the configuration space of the object, which critically penalises the performance of the planner (see Figures 5.1 and 5.2). The same figures shown that varying the size of

options in terms of possible actions affects the ability of the algorithm to converge to the goal region (see Sec. 5.4).

A special note should also be made for the normalising factor L. I compute L as the maximum pairwise translational distance, such that the rotational and translational terms in the cost function have the same magnitude (see Eq. 5.4). This choice biases the exploration strategy in favour of reducing the rotational error. In the experimental results, the terminal conditions mirror this bias, allowing us to relax the tolerance on the (final) translational error: 6 cm versus 9 degrees of tolerance for rotational errors. However, Fig. 5.1(a) shows that these are not the asymptotically achieving errors. In fact the planner reaches the required translational accuracy terminal condition, on average, with half of the total nodes in its final path, after this, even though the planner is no longer encouraged to reduce the translation error, the final error achieved is around 4 cm. Additionally, the convergence properties of RRTs to a goal region with an arbitrary tolerance have been intensively studied (see e.g. [LaValle, 2006]), showing that the algorithm is probabilistically complete: "It is not difficult to prove that the vertices will become uniformly distributed. As the RRT initially expands, the vertices are clearly not uniformly distributed; however, the probability that a randomly-chosen point lies within Δt of a vertex of the tree eventually approaches one." [LaValle, 1998].

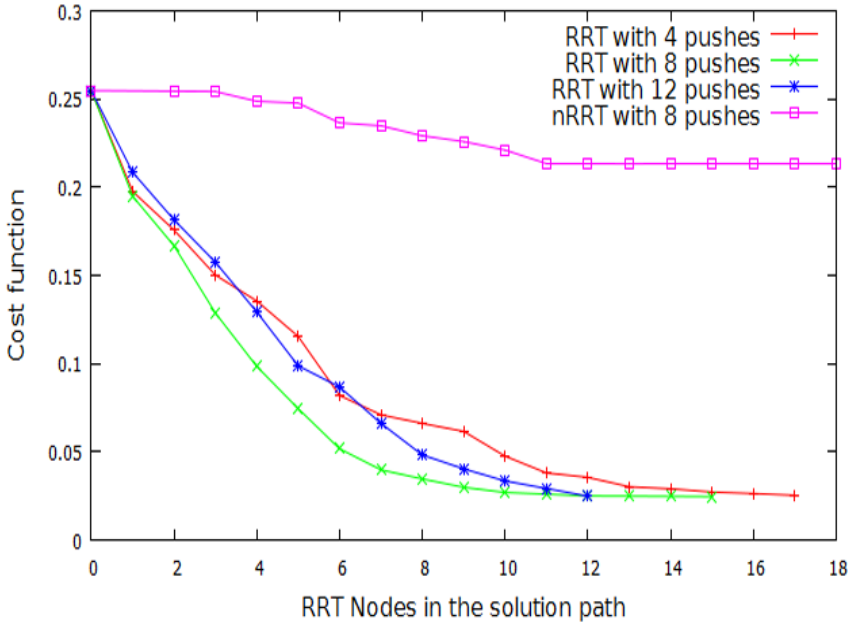


(a)

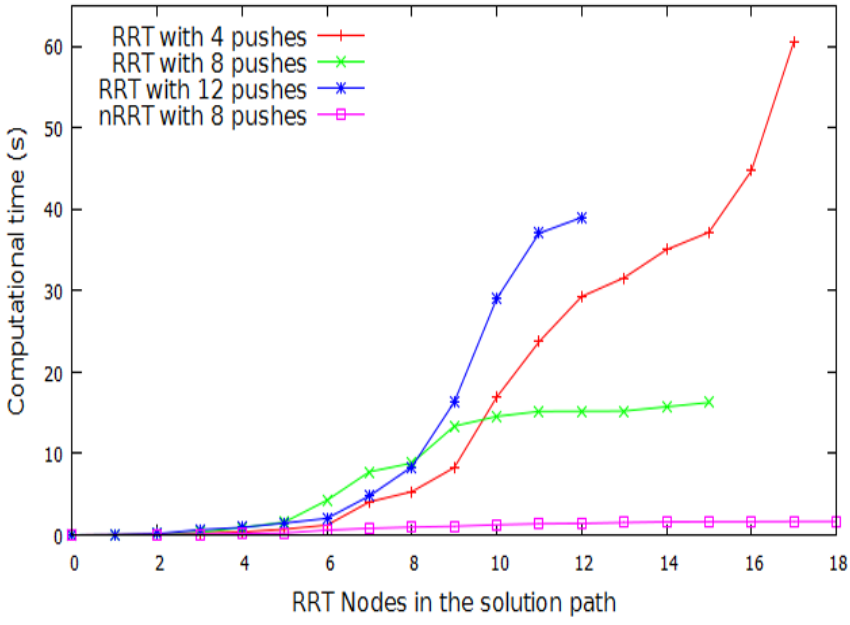


(b)

Figure 5.1: Charts (a) and (b) plot the final translational and rotational errors for solution plans with three different numbers N of random push choices. The graphs show how the mean costs (over ten trials) vary as successive additional nodes are added to the RRT trees. The x-axis shows the number of RRT nodes which compose the solution path. The results are compared with a "naive RRT" which executes only a single push to move from one RRT node to the next ($\zeta_2 = 1$). We let the "naive RRT" explore $N = 8$ possible push directions, since this appears optimal compared to 4 or 12 direction choices. The rotational error in quaternion sphere is bounded in $[0, 1]$ where 1 means 90 degrees of error, thus the terminal condition of 9 degrees is equal to 0.01 in 5.1(b).



(a)



(b)

Figure 5.2: Charts (a) show overall cost (combined rotation and translation error), (b) shows computation time for solution plans with three different numbers N of random push choices. The graphs show how the mean costs (over ten trials) vary as successive additional nodes are added to the RRT trees. The x-axis shows the number of RRT nodes which compose the solution path. The results are compared with a "naive RRT" which executes only a single push to move from one RRT node to the next ($\zeta_2 = 1$). We let the "naive RRT" explore $N = 8$ possible push directions, since this appears optimal compared to 4 or 12 direction choices.

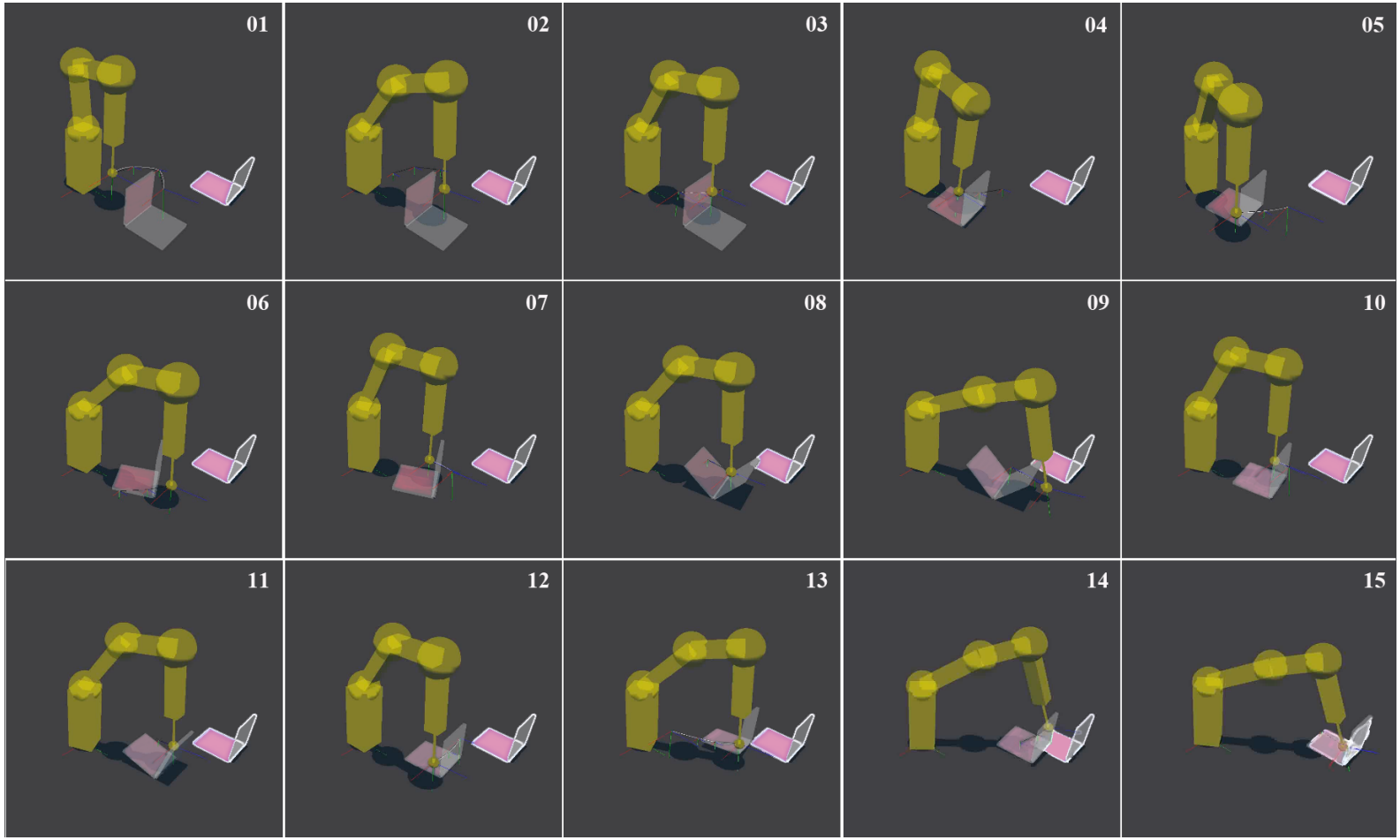


Figure 5.3: The image sequence shows a solution path computed by the planning algorithm in which 8 different pushes were randomly selected at each iteration. Image 01 shows the initial pose. The wire-framed L-shaped object (or polyflap) is a 'phantom' to indicate the desired goal state. The goal pose is translated from the initial pose by 28 cm and rotated by 90 degrees. In this trial the algorithm has planned a distinctly different strategy from the one shown in Fig. 1.3. Image 02 shows the collision-free trajectory to bring the end effector to the start pose of the first push. Images 01-04 show the first push which makes the polyflap tip over. Images 05-09 show a series of pushes which culminate in the polyflap resting in an unstable equilibrium pose along its folded edge. Images 12-13 show a sideways push. Images 14-15 show the final frontal push.

Chapter 6

Path planning for informational effects: pose uncertainty

Chapter 4 has shown that decision-theoretical frameworks can be extended to domains in which the agent has no perfect sensing abilities. This is a typical situation in robotics, where a robot perceives the world through noisy sensors. To perform tasks in such environments, a robot has to be capable of gathering information to recover its knowledge of the world, if necessary.

This chapter shows how to reason about information effects to refine the localisation of the object to be grasped when the object is not in the expected pose. The main novel outcome is thus to enable tactile information gain planning for dexterous, high degree of freedom (DoF) manipulators. This is achieved by using a combination of information gain planning, hierarchical probabilistic roadmap planning, and belief updating from tactile sensors for objects with non-Gaussian pose uncertainty in 6 dimensions.

In this chapter it will be assumed that a single object with a known shape model is available as well as a desired robot grasp configuration with respect to the object. This

means that only the pose of the object is uncertain. In addition, the algorithm will treat subsequent reach-to-grasp problems as independent. This means that the robot manipulator will be withdrawn to a safe position before re-planning the next reach-to-grasp trajectory. In the next chapter, we will discuss how to relax some of these assumptions.

In this chapter, the method is demonstrated in trials with simulated robots. Sequential re-planning is shown to achieve a greater success rate than single grasp attempts, and trajectories that maximise information gain require fewer re-planning iterations than conventional planning methods before a grasp is achieved.

6.1 Organisation

This chapter proceeds as follows. Section 6.2 presents the problem of planning a reach-to-grasp trajectory for dexterous robot manipulators under uncertainty on the pose of the object to be grasped. A main contribution of this thesis is that of building solutions that simultaneously attempt to perform a task (here dexterous grasping) as well as gather information (i.e. locate the object to be grasped), if necessary.

Section 6.3 presents an alternative formulation for the problem of planning grasping under uncertainty. Instead of formalising it as an ISMDP problem, I pose the problem as an optimisation problem.

Section 6.4 discusses the implementation details of my proposed approach.

Section 6.5 presents experimental test results in which a dexterous grasping trajectory has to be planned under non-Gaussian object-pose uncertainty in 6D. Three strategies are presented.

Section 6.6 summarises the contributions of this chapters.

6.2 Introduction

As described in Sec. 1.3.2, robot grasping is typically affected by uncertainty associated with the location of the object to be grasped. This is a challenging problem because requires us to find collision-free trajectories in a high-dimensional space (the robot's joint space, Sec. 3.2.3) as well as to reason about how to act more robustly in the face of such an uncertainty. Section 2.5 has reviewed previous work on planning dexterous grasping under object pose uncertainty. Typically these efforts are based on the separation of information gathering trajectories from grasping trajectories.

The main contributions of this work are:

- Planning information gain trajectories whilst simultaneously attempting to grasp the target object.
- Using a hierarchical sample-based path planner, here a Probabilistic Roadmap (PRM) planner, which encodes expected information gain (in a low-dimensional belief space) in its trajectory segments.
- Refining the expected object pose by using an observation model for contact sensing by a multi-finger hand that palpates a 3D object to be grasped.
- Showing that this approach enables planning for robot manipulators with 21 DoF and non-Gaussian object pose uncertainty in 6 dimensions.

Several assumptions are made. First, the object is of known shape, in the sense that a high density point cloud model or mesh model is available. Second, a pre-computed grasp (i.e. a set of finger contacts) and its pre-grasp configuration are known a priori for this object. Third, the algorithm assumes the availability of an unreliable estimate of the object's pose. In addition, each re-planning iteration is treated as independent and thus the robot's manipulator was withdrawn to a safe position after a failed attempt to

grasp, and the (tactile) sensing abilities of the robot are assumed to be perfect (no false detections).

In this scenario (shown in Fig. 6.1), a depth image is obtained from an Asus Xtion Pro depth camera. This gives an incomplete point cloud of the object surface, I refer to this new point cloud as a *query point cloud* or simply *query* to distinguish it from the point cloud used as the *model*. Using a model fitting procedure similar to the sampling from surflet pairs method presented in [Hillenbrand & Fuchs, 2011], a probability density (or belief state) over the object pose is estimated, represented as a particle set. Given this distribution, a reach-to-grasp trajectory is planned. This trajectory has as its goal configuration the pre-computed grasp under the assumption that the object is at a pose corresponding to the mean position of the particle set. The path to this goal configuration is found using a stochastic motion planner. This planner works with a cost function that allows deviations from the shortest path that maximise the chance of gathering tactile observations that will reduce pose uncertainty in the object location. If unexpected observations occurred during the execution of the planned trajectory, then such observations (both tactile contact and no-contact signals) collected at poses along the reach-to-grasp trajectory are used to perform a particle filter update. Re-planning then occurs with the new pose distribution, and a new reach-to-grasp trajectory can be constructed. This process is then iterated until a successful grasp is achieved.

In Sec. 4.4.3, we discussed the benefits of planning in belief spaces as well as its limitations in real world applications. Thus, this work is related to the approach of the approach of Platt et al. [Platt et al., 2001], [Platt et al., 2012] (discussed in more details in Sec. 2.5). That work plans a sequence of actions that will generate observations that distinguish a hypothesised state from competing hypotheses while also reaching a goal position. The informational value of a trajectory is the difference in the expected observations between the hypothesised position and each alternative. This approach allows us to

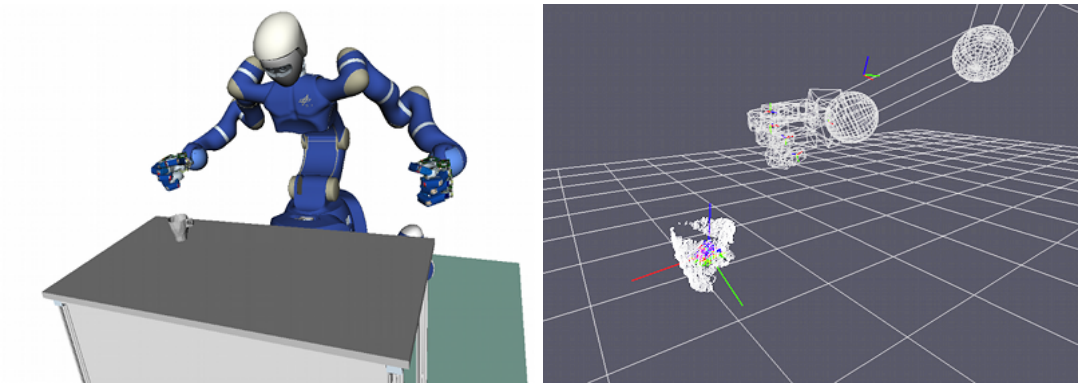


Figure 6.1: Justin and a known mug to grasp in OpenRave simulator (left). Partial point cloud of the mug used for localisation (right).

track high-dimensional belief states using an accurate filter defined by the user, but reduces the complexity of planning in belief spaces by approximating the informational value of actions from a low-dimensional subspace of the belief state. Platt et al. applied this to planning for a two degree of freedom manipulator using a laser range finder for observations, and employed an optimisation framework for planning. The algorithm is proved to localise the true state of the system in 1 dimension and to reach a goal region with high probability. In contrast to [Platt et al., 2001], my approach encodes information gathering actions to localise an object to be grasped in 6 dimensions while simultaneously attempting to achieve the task of grasping. Similarly to [Platt et al., 2001], [Platt et al., 2012], this method is guaranteed to converge to the true state of the system in which a reach-to-grasp trajectory succeeds with high probability, under the assumption that the system is not perturbed by previous grasping attempts (e.g. the robot contacts the target object without changing its configuration). Here we also discuss how this approach can be extended to planning the motion of a manipulator performing multi-finger grasping.

This approach has been tested in simulation, as described in Sec. 1.4.2. The following section introduces the formulation of the problem of planning grasping under uncertainty used in this thesis.

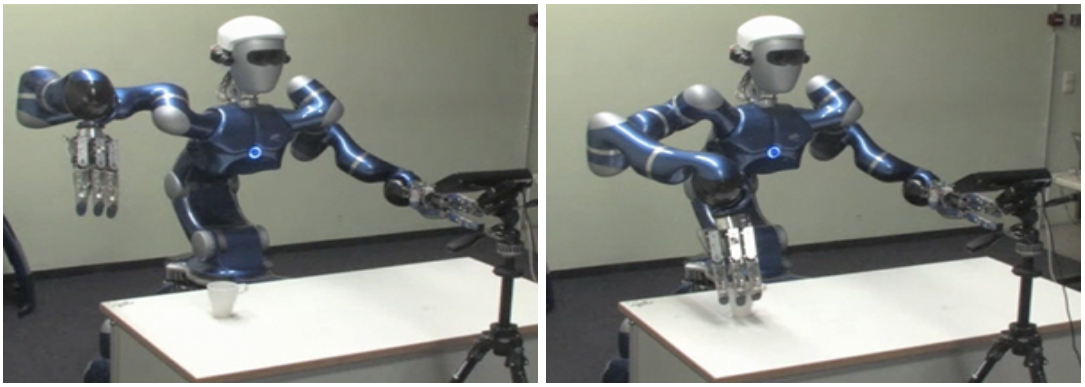


Figure 6.2: Rollin Justin at DLR. Justin in a starting configuration, the mug to be grasped (glued on the desk) and the depth camera (left). Justin executing a successful reach-to-grasp trajectory which leads to grasp the mug (right). Since the mug is glued on the desk, Justin cannot lift it up to prove the real stability of the grasp.

6.3 Planning trajectory

This chapter presents a formulation to the problem of planning dexterous grasping trajectories under object pose uncertainty using information gain re-planning. First I show how tactile information, acquired during a failed attempt to grasp an object can be used to refine the estimate of that object's pose, and then how this information can be used to re-plan new reach-to-grasp trajectories for successive grasp attempts. Finally I show how reach-to-grasp trajectories can be modified, so that they maximise the expected tactile information gain, while simultaneously delivering the hand to the grasp configuration that is most likely to succeed.

6.3.1 Problem formulation

This chapter is concerned with the problem of planning control actions to reach a goal state in the presence of incomplete or noisy observations. Let us consider a discrete-time system with continuous non-linear deterministic dynamics,

$$x_{t+1} = f(x_t, u_t)$$

where $x_t \in \mathbb{R}^n$ is a configuration state of the robot and $u_t \in \mathbb{R}^l$ is an action vector, both parametrised over time $t \in \{1, 2, ...\}$. Let $p \in SE(3)$ describe the object pose, given an initial prior belief state b_1 and let us define a set of k hypotheses as $\{p^i\}_{i=1}^k$, where $p^1 = \arg\max b_1$ and $p^i \sim b_1, i \in [2, k]$. Hence we ensure that p^1 is the maximum likelihood (ML) pose given the initial belief state b_1, while the rest of the hypotheses are sampled from the belief state. The idea is to search for a sequence of actions, $u_{1:T-1} = \{u_1, \ldots, u_{T-1}\}$, that distinguish between observations that would occur if the object were in p^1 from any other p^i pose, with $i \in [2, k]$. At each time step, t, the system makes an observation, $y \in \mathbb{R}^m$, that is a non-linear stochastic function of states and hypotheses. Without losing generality, we define y_t to be a column vector of binary values. Each of these values represents whether or not a contact is observed between a given link of the robot and the hypothesis p^i. However, binary values have been shown to be not very informative during the planning phase. Therefore let us define,

$$h(x, p^i) = Pr(y = 1|x, p^i)$$

as a column vector of scores identifying the likelihood of observing a contact, $y = 1$, as function of states and hypotheses (more details in Sec. 6.4.1). More generally, let $F_t(x, u_{1:t-1})$ be the robot configuration at time t if the system begins at state x and takes action $u_{1:t-1}$. Therefore the expected sequence of observations over a trajectory, $u_{1:t-1}$, is:

$$h_t(x, u_{1:t-1}, p^i) = (h(F_2(x, u_1), p^i)^T, h(F_3(x, u_{1:2}), p^i)^T, \ldots, h(F_t(x, u_{1:t-1}), p^i)^T)^T$$

a column vector which describes the likelihood of observing a contact at any time step of the trajectory $u_{1:t-1}$. We then need to search for a sequence of actions which maximise the difference between observations that are expected to happen in the sampled states, $p^{2:k}$, when the system is actually in the most likely hypothesis, p^1. In other words, we

want to find a sequence of action, $u_{1:T-1}$, that minimises

$$\mathcal{J}(x, u_{1:T-1}, p^{1:k}) = \sum_{i=2}^{k} N(h_t(x, u_{1:T-1}, p^i) | h_t(x, u_{1:T-1}, p^1), \mathbb{Q}) \tag{6.1}$$

where $N(\cdot|\mu, \Sigma)$ denotes the Gaussian distribution with mean μ and covariance Σ and $\mathbb{Q}$ is the block diagonal of measurement noise covariance matrices of appropriate size. Specifically to this application, $\mathbb{Q} \in \mathbb{R}^{6x6}$ is the standard deviation of the sampled poses (particles) that compose the belief state in terms of the 6 dimensions that determine position and orientation. Rather than optimising (6.1) we follow the suggested simplifications described in [Platt et al., 2001] by dropping the normalisation factor in the Gaussian and optimising the exponential factor only. Let us define for any $i \in [2, k]$

$$\Phi(x, u_{1:T-1}, p^i) = ||h_t(x, u_{1:T-1}, p^i) - h_t(x, u_{1:T-1}, p^1)||^2_{\mathbb{Q}},$$

then the modified cost function is

$$\mathcal{J}(x, u_{1:T-1}, p^{1:k}) = \frac{1}{k} \sum_{i=2}^{k} e^{-\Phi(x, u_{1:T-1}, p^i)} \tag{6.2}$$

it is worth to notice that when there is a significant difference between the sequence of expected observations, $h_t(x, u_{1:T-1}, p^i)$ and $h_t(x, u_{1:T-1}, p^1)$, the function $\Phi(\cdot)$ is large and therefore $\mathcal{J}(\cdot)$ is small. On the other hand if the sequence of expected observations are very similar to each other, their distance measurement tends to 0 and $\mathcal{J}(\cdot)$ tends to 1. Equation (6.2) can be minimised using different planning techniques such as Randomly-exploring Random Trees (RRTs) [LaValle, 1998], Probabilistic Roadmap (PRM) [Kavraki & Svestka, 1996], Differential Dynamic Programming (DDP) [Jacobson & Mayne, 1970] or Sequential Dynamic Programming (SDP) [Betts, 2001]. In Section 6.4.2, we define a new set of heuristics that can be encoded in a PRM planner for generating more reliable trajectories that explicitly reason over the pose uncertainty, and

demonstrate the method with experiments on a virtual testbed.

6.3.2 Mean pose estimate

As mentioned before, the sequential re-planning approach presented in this thesis assumes the availability of an unreliable estimate of the object's pose to construct a grasping trajectory. In Sec. 2.3 we discussed the advantages of encoding states as probability distributions, rather than relying on ML estimations. In particular, we discussed the benefits of using a particle filter algorithm, as representation of the belief space, to cope with multi-modal uncertainty. For these reasons, the approach presented in this thesis is to estimate a set of particles (or object-pose hypotheses) in a PF fashion from real RGB-D data. Each particle is the result of an ML estimation and it can be considered as a rigid body transformation which maximises the likelihood of best aligning the (dense) model point cloud (assumed to be available in the system) to the (partial) query point cloud.

Note that this rigid body transformation is the result of a sampling-based model-fitting procedure similar to the one presented in [Hillenbrand & Fuchs, 2011]. This procedure samples a random pair of features from the query point cloud (such as a pair of points with their relative normals) and computes the rigid body transformation to the closest pair of features in the model. A mean-shift algorithm is used to compute the most likely transformation from model to query. The mean-shift algorithm is an optimisation procedure, which iteratively seeks for the mode of a density function given a set of discrete samples from that distribution [Cheng, 1995]. The ML estimation is biased by the set of sampled pairs of features. For this reason, for each particle a new set of features is constructed to ensure that the ML estimations are distinct one from the other.

Once this set of particles is computed, it is possible to calculate the object's pose estimate by again using a mean-shift algorithm on the particle set.

6.3.3 Belief update

After a trajectory is planned the robot executes it. If an unexpected observation occurs at execution-time the algorithm refines the current belief state using an accurate, high-dimensional filter provided by the user.

In order to define an unexpected observation, it may be convenient for the reader to think of a reach-to-grasp trajectory as composed of two parts: i) approaching trajectory which leads to a pre-grasp configuration of the robot in which the fingers generally cage the object to be grasped without generating any contact, and ii) a grasping trajectory which moves the fingers into contact and eventually generates a force closure grasp. In this way any contact which occurs during the approaching trajectory is considered as an unexpected observation. Similarly a non sufficient number of contacts for a force closure at the end of a grasping trajectory is considered as unexpected and can also be used as an observation.

In this implementation, the belief is updated using the Bayes rule assuming deterministic dynamics. In this case we can write the belief update as

$$b_{t+1} = \frac{Pr(y_{t+1}|b_t, u_t)b_t}{Pr(y_{t+1})} \tag{6.3}$$

6.3.4 Re-planning

The sequential re-planning algorithm plans trajectories assuming only the maximum likelihood observations given the current belief state. Therefore the robot must rely on sensory feedback during the execution of the planned trajectory in order to detect whether or not unexpected observations occur. This triggers a belief update, using the observation gathered at execution-time, and consequently a re-planning phase, in which the manipulator is moved back to a safe configuration (e.g. outside the uncertain region)

and a new reach-to-grasp trajectory is planned.

In the experiments, the algorithm uses torque sensors at each joint of the robot's hand to detect whether or not a link of the hand is in contact with the environment. It is assumed that the sensing abilities of the robot are fine enough to perceive the object without moving it. Even in simulation it is difficult to maintain such an assumption, however the results suggest that small changes in the configuration of the object do not affect the algorithm which is still able to converge to a force closure grasp.

6.3.5 Terminal conditions

The algorithm terminates its execution when i) no unexpected observations are detected and, ii) the robot achieves a force closure grasp which satisfies a user-defined (minimum) grasp quality. In simulation, knowing the model of the object, the contact points and the executed forces, it is possible to make a force closure analysis using an associated quality measure [Suarez et al., 2006]. In this work, the magnitude of the minimum perturbation wrench that breaks the force closure is used as the grasp quality measure [Ferrari & Canny, 1992].

A possible limitation of this implementation is that the unexpected observations are treated as binary input (contact, no contact). In other words, a contact in the approaching trajectory will always trigger re-planning, even if the contact occurred at the very last step of the trajectory and the robot's end-effector is in a configuration where it is still possible to achieve the target grasp. We aim to investigate such cases in future work.

6.4 Implementation

The approach presented here consists in planning informative tactile observations for a dexterous hand while simultaneously reaching a given target grasp. The main innovations are:

- Implementing a hierarchical PRM algorithm which allows us to plan dexterous reach-to-grasp trajectories;
- Encoding a new set of heuristics for a randomised motion planner;
- Formalising an observational model for contact sensing by a multi-finger hand.

This approach has been empirically tested in a simulated environment for robotic manipulators up to 21 DoF under non-Gaussian object pose uncertainty in 6 dimensions.

6.4.1 Observational model

Let us assume that a robotic manipulator is composed of parts. These parts are considered collections (or chains) of joints linked somehow together. Without losing generality, we also assume a single part called 'arm' and a set of $\mathbf{M}$ parts called 'fingers'. In addition, I describe the observational model as limited only to a given subset of those parts, i.e. only fingers or a subset of them (e.g. finger tips). Let $\mathbf{M}$ be the ordered set of parts which compose the manipulator, then $x(j)$ is the configuration in joint space of the j^{th} part, with $j \in M$. In other words, j is the index of a specific chain. We also define $\bar{\mathbf{M}}$ to be the set of indices such that the respective part is used in the observational model. In addition, we use the operator $\mathcal{W}(x(j))$ to refer to the workspace coordinates in $SE(3)$ of the j^{th} joint with respect to a given reference frame.

Mathematically I formalise the likelihood of observing a contact for each finger of the robot as an exponential distribution over the Euclidian distance, d_{ji}, between the finger

tip's pose, $\mathcal{W}(x(j))$, and the closest surface of the object assumed to be in pose p^i. Note that, within the scope of this work, the observational model is limited to contacts which may occur on the internal surface of fingers. This directly affects the planner which rewards trajectories that would generate contacts on the finger tips rather than on the back side of the fingers. Therefore for any $j \in \bar{M}$ we write

$$Pr(y(j) = 1|x(j), p^i) = \begin{cases} \eta \exp(-\lambda d_{ji}) & \text{if } d_{ji} \leq d_{max} \\ & \text{and } \langle n_{xj}, \hat{n_{p^i}} \rangle < 0 \\ 0 & \text{otherwise} \end{cases}$$

where $\langle n_{xj}, \hat{n}_{p^i} \rangle$ is the inner product of, respectively, j^{th} finger tip's normal and the estimated object surface's normal, and d_{max} describes a maximum range in which the likelihood of reading a contact is not zero, η is a normaliser. Note that in other work the likelihood of observing a contact has been modelled as Gaussian distributions (see e.g. [Hsiao et al., 2011]), however any monotonic decreasing function would be suitable for this purpose. That allows us to rewrite the likelihood of reading a contact on the force/torque sensors of the robot, $h(x, p^i), i \in [1, k]$ with $j_1, \ldots, j_m \in \bar{M}$ as follows,

$$h(x, p^i) = [Pr(y(j_1) = 1|x(j_1), p^i), \ldots, Pr(y(j_n) = 1|x(j_m), p^i)]^T$$

6.4.2 Planning a trajectory to maximise information gain

The implementation of this planner uses a modified version of Probabilistic Roadmap (PRM), [Kavraki & Svestka, 1996], to plan trajectories and detect collisions. As discussed in Sec. 3.3.2.2, the PRM algorithm is composed of two phases: i) *learning phase*, in which a connected graph **G** of obstacle-free configurations of the robot is generated and, ii) *query phase*, in which a path is searched for a given pair of configurations x_{root}, x_{goal}.

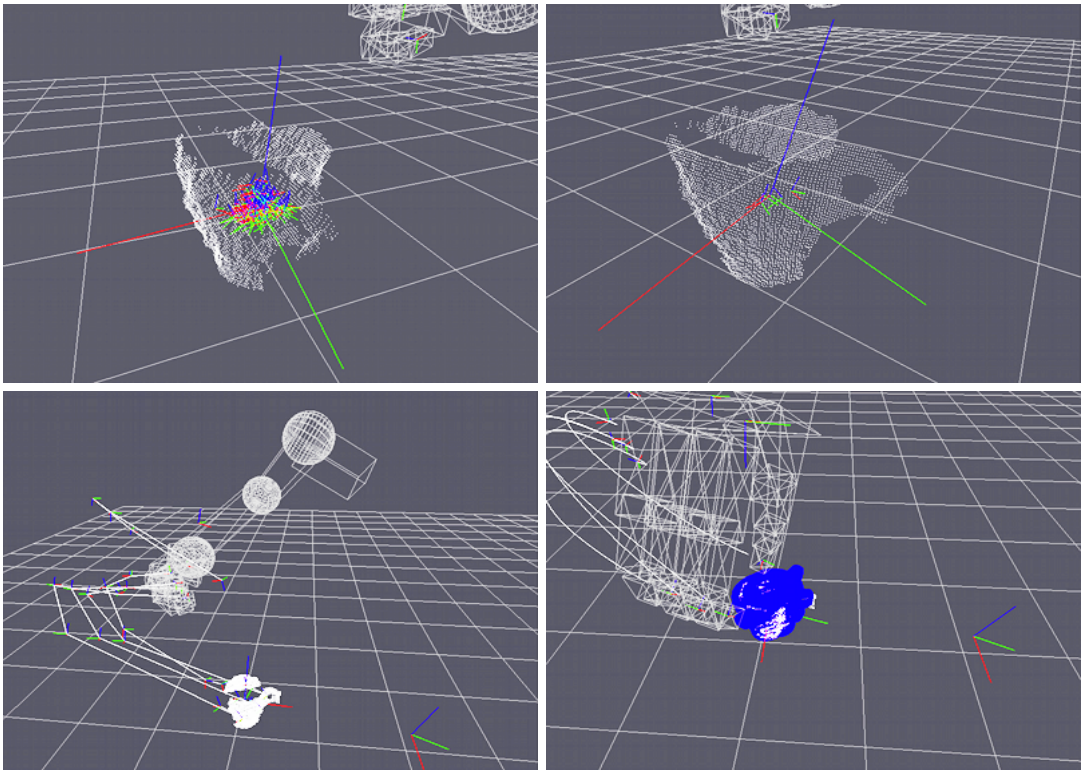


Figure 6.3: Top row: The high dimensional belief state used to track object pose (left); the low dimensional filter used for planning (right). Bottom row: The unoptimised PRM plan for fingers and wrist (left); the optimised and smoothed trajectory (right).

However the computational cost for the learning phase grows very fast with respect to the dimensionality of the problem. This planner therefore incrementally builds connections between neighbouring nodes during the query phase. Given a pair $\langle x_{root}, w_{goal} \rangle$ which describes the root state in configuration space, $x_{root} \in \mathbb{R}^n$, and the goal state in workspace, $w_{goal} \in SE(3)$, of the trajectory, this planner uses an A* algorithm to find a minimum cost trajectory in obstacle-free joint space according to:

$$c(x) = c_1(x, x_{root}) + c_2(x, x', \hat{x}_{goal}) \tag{6.4}$$

where $x, x' \in \mathbb{R}^n$ and $x' \in Neighbour(x)$, $\hat{x}_{goal}$ is a reachable goal configuration for the robot computed by inverse kinematics, $c_1(x, x_{root})$ is the cost-to-reach x from x_{root} and

$c_2(x, x', \hat{x}_{goal})$ is a linear combination of the cost-to-go from x to a neighbour node x' and the expected cost-to-go from x' to the target. I implemented $c_1(\cdot)$ as a cumulative sum of travelled distances. More specifically, I define

$$\begin{aligned} c_2(x, x', \hat{x}_{goal}) = \alpha d_{bound}(x, x') + \beta d(x', \hat{x}_{goal}) \\ + \gamma d_{cfg}(x) \end{aligned} \tag{6.5}$$

where $\alpha, \beta, \gamma \in \mathbb{R}$, $d(\cdot)$ is a distance function in $SE(3)$ which linearly combine rotational and translational distances in workspace[1]. The function $d_{bound}(\cdot)$ behaves as $d(\cdot)$ for neighbouring nodes, but returns $+\infty$ otherwise. Two nodes are considered neighbours if their linearly combine distance is less than a user-defined threshold (in this thesis this value is kept fixed to 0.01). Finally, $d_{cfg}(\cdot)$ is a function which penalises dangerous configurations of the robot (i.e. close to joint limits).

I redefine the heuristic $c_2(\cdot)$ in order to reward informative tactile explorations while attempting to reach the goal state (described as a target configuration of the manipulator).

$$\begin{aligned} \bar{c}_2(x, x', \hat{x}_{goal}, A, p^{1:k}) = &\alpha \mathcal{J}(x, x', p^{1:k}) d_{bound}(x, x') \\ &+ \beta d_A(x', \hat{x}_{goal}) + \gamma d_{cfg}(x') \end{aligned} \tag{6.6}$$

where A is the diagonal covariance matrix of the sampled states, for any column vector $a, \mu \in \mathbb{R}^n$, $d_A(a, \mu) = \sqrt{(a-\mu)^T A^{-1} (a-\mu)}$ is the Mahalanobis distance centred in μ and $\mathcal{J}(x, x', p^{1:k}) \in (0, 1]$ is a factor which rewards trajectories with a large difference between expected observations if the object is at the expected location, p^1, versus observations that would be expected if the object is at other poses, $p^{2:k}$, sampled from the distribution of poses associated with the object's positional uncertainty:

$$\mathcal{J}(x, x', p^{1:k}) = \frac{1}{k-1} \sum_{i=2}^{k} e^{-\Phi(x, x', p^i)} \tag{6.7}$$

[1]For the sake of simplicity, I reduce the mathematical notation by writing $d(x, x')$ instead of d(W(x),W(x')).

where:

$$\Phi(x, x', p^i) = ||h_t(x, x', p^i) - h_t(x, x', p^1)||_2$$

for each $i \in [2, k]$ and $h_t(x, x', p^i)$ is sequence of probability of reading a contact travelling from state x to x'. In this implementation $h_t(x, x', p^i) = h(x', p^i)$. In other words, I evaluate the likelihood of making a contact while moving from state x to x' as the likelihood of making a contact only in the next state x'. Note that this observational model is designed to conserve (6.6) as in (6.5) when the likelihood of observing a tactile contact is zero. In fact, for robot configurations in which the distance to the sampled poses is larger than a threshold, d_{max}, the cost function $\mathcal{J}(\cdot)$ is equal to 1. However, by employing the Mahalanobis distance in $d_A(\cdot)$, I also encode uncertainty in the second factor of the heuristics, which evaluates the expected distance to the goal configuration. In this way the planner also copes with pose uncertainty at the early stages of the trajectory, when the robot is still too far away from the object to observe any contacts.

It is important to notice that the heuristic in Eq. 6.6 does not necessarily need to be an admissible heuristic for the A* algorithm. This is due to the fact that we are interested in finding the most informative trajectory rather that the shortest one, hence it is in our interest to allow the planner of deviating from the shortest path in order to maximise the expected information gain.

6.4.3 Planning for Dexterous manipulator

In order to compute a dexterous trajectory which allows us to plan movement for both arm and fingers we need to break down the curse of dimensionality or, equivalently, increase the number of sampled configurations to properly cover the configuration space.

The proposed solution is to build a hierarchical planner. First a PRM is constructed only

in the arm configuration space in order to find a global path between the $x_{root}, \hat{x}_{goal}$. It is worth noticing that in this phase the rest of the joints of the manipulator are interpolated in order to have a smooth passage from x_{root} to $\hat{x}_{goal}$. Then the planned trajectory is refined by constructing a new PRM in the entire configuration space of the manipulator (e.g. arm + hand joint space) along the global path. In other words, this approach limits the new PRM to explore only the subspace nearby the configurations which compose the global path. Subsequently an optimisation procedure is executed along the trajectory to generate a smoother transition from one configuration to the next.

This approach enable us to plan dexterous reach-to-grasp trajectories up to 21 DoF with only 1,000 sampled configurations. Note that this is the same order of magnitude that it is used in practise for planning trajectories of much simpler 6 DoF robot manipulators.

6.4.4 Belief update

Once a trajectory is executed and a real (unexpected) observation y is detected, the belief state is updated according to the Bayes' rule. The belief state is represented as a set of N particles $b_t = \{b_t^z\}_{z=1}^N$. In a particle filter fashion the weight of each particle b_t^z, $z \in \{1, \ldots, N\}$ is updated as follows

$$Pr(y|x, b_t^z) = \prod_{j \in \hat{M}} Pr(y_t(j)|x_t(j), b_t^z)$$

and then re-sampling is performed to generate a posterior distribution b_{t+1} as new set of particles $\{b_{t+1}^z\}_{z=1}^N$.

In simulation this approach assumes that there are no false detections. However it is possible to distinguish whether or not a contact occurs between the object to be grasped and the robot's end-effector. For example, in case a contact with the environment is

Table 6.1: Experimental results in simulation. The highlighted entry shows an interesting case in which the informational gain planning (ReGrasp+IG) grasps at the first iteration while ReGrasp converges to a grasp in 7.

Initial error		Single	ReGrasp		ReGrasp+IG	
Lin (m)	Ang (quat)		Iterations	FCA	Iterations	FCA
0.055909	0.006344	success	1	0.006301	2	0.005876
0.05343	0.012268	success	1	0.006072	2	0.00649
0.057804	0.013443	success	1	0.005996	2	0.005308
0.060809	0.016942	failure	2	0.000452	1	0.000911
0.058412	0.017696	success	1	0.008286	1	0.006266
0.05996	0.019115	failure	4	0.005679	1	0.008111
0.05755	0.019915	success	1	0.005936	1	0.003597
0.05815	**0.020103**	**failure**	**7**	**0.003868**	**1**	**0.003737**
0.059598	0.021463	failure	3	0.007579	1	0.006105
0.061404	0.023431	success	1	0.007397	1	0.006409
0.063758	0.025339	success	1	0.007959	2	0.002738
0.059935	0.054883	failure	2	0.005933	2	0.00752

detected, the algorithm skips the belief update step and moves the robot back to a safe configuration before triggering the re-planning.

6.5 Results

In this work I aim to show that sequential re-planning is capable of achieving higher successful grasp rates than single grasp attempts, in presence of non-Gaussian object-pose uncertainty in 6 dimensions. I also show that planning trajectories that maximise information gain requires fewer re-planning iterations to achieve a grasp with the same order of magnitude of grasp quality.

The algorithm has been tested by running 12 trials in a virtual environment. Each trial has a different initial probability density over the object pose, so to test the ability of different strategies to achieve a grasp configuration in which it is possible to obtain force closure grasp despite the pose uncertainty. In these experiments, after an unexpected

observation occurs, the simulated robot is always moved back to the initial configuration, which reproduces the same pose of the real robot shown in Fig. 6.2 (left image).

Table 6.1 summarises the data collected in the experiments. First the algorithm computed the error in the initial estimation of the object pose with respect to the ground truth, which is available in the simulation environment. This error is decomposed into translational and rotational components. The translational component measures displacement as Euclidian distance in a 3 dimensional space between the estimated location and the ground truth. The rotational or angular displacement is evaluated using the quaternion representation.

Next, three different algorithms are performed: i) an open-loop trajectory towards the expected object pose without re-planning, ii) a sequential re-planning algorithm, which exploits contact observations gathered during the previous grasp attempt, but does not plan trajectories specifically to maximise information gain (ReGrasp) and iii) sequential re-planning with reach-to-grasp trajectories which are specifically optimised to maximise the expected tactile information gain, while also achieving the desired grasp configuration (ReGrasp+IG). Column 3 in table 6.1 shows the rate of successful attempts for the open-loop trajectory. Columns 4 and 5 show that successive re-planning enables us to achieve a grasp in cases when the open-loop strategy fails. The table presents both the number of (re-)planning iterations and the grasp quality value once a force closure grasp is achieved. Finally, columns 6 and 7 show that, planning trajectories which maximise information gain, reduces the number of re-planning iterations required to achieve a successful grasp while producing the same order of magnitude of grasp quality. Note that, for all the trials, the nominal grasp quality computed on the object in the nominal pose is 0.006920.

An interesting case is shown in the 8th row of Table 6.1. In this trial a single attempt fails while ReGrasp requires 7 iterations to generate a quite poor force closure. Instead

using ReGrasp+IG produces a successful trajectory at the very first attempt, although the grasp quality does not improve in this specific case. This trial clearly shows that the ability of reasoning about information gain allows ReGrasp+IG to produce more robust reach-to-grasp trajectories with respect to pose uncertainty, in fact the other planners have failed to generate a successful grasp at the first iteration.

To illustrate the behaviour of the re-planning system, Fig. 6.4 shows a typical sequence generated by one simulation trial. The algorithm assumes a known point cloud model of the object shape, and uncertainty in the object pose. In this trial the robot observes the object as a point cloud and applies a model fitting process described in [Hillenbrand & Fuchs, 2011]. The model fitting process is stochastic and so the resulting pose of the object is uncertain. Multiple poses are sampled by repeating this process, and to obtain a belief state consisting of the resulting set of possible poses of the object. A trajectory is then planned to achieve a given grasp on the object. At each step a trajectory for the wrist and fingers are generated that will move to the desired grasp, while deviating from a minimum length trajectory to maximise information gathered through tactile observations. The belief state is updated and re-planning occurs each time a tactile contact is made. In this example three separate contacts are made during reach-to-grasp trajectories. In each instance the trajectory is re-planned given the new belief state. After three contacts the fourth trajectory achieves a configuration suitable for grasping.

A drawback of ReGrasp+IG is the computational time. In the experiments ReGrasp requires on average ~ 7 seconds to plan a trajectory as compared to ~ 200 seconds for ReGrasp+IG. ReGrasp+IG adds extra computation during the query phase of the PRM algorithm in order to compute the likelihood of reading a contact, which requires finding the closest surface to the finger tips for each hypothesis (represented as point clouds) every time a node in the PRM is in a neighbourhood of the uncertain region. In the next

chapter, I will present a fast and efficient KD-tree based algorithm for collision detection for non-convex objects represented as point clouds (Sec. 7.3.5) to overcome this extra computation in the planning phase.

6.6 Conclusion

In this chapter I have shown how to solve the problem of dexterous grasping of objects with uncertain pose by using information gain re-planning.

I have proposed a method for tactile information gain planning for dexterous, high DoF manipulators by using i) a hierarchical PRM planner which encodes informational measure in its segments, and ii) a method for sequentially refining pose uncertainty, by using tactile observations gathered during unsuccessful grasp attempts. This approach enables planning for robot manipulators with 21 DoF and non-Gaussian object pose uncertainty in 6 dimensions.

I have also shown how sequential re-planning can achieve better quality grasps than single attempts to directly grasp an object at its expected pose, and that re-planning with trajectories designed to maximise tactile information gain, achieves successful grasps with fewer iterations than sequential attempts to grasp directly towards the object's (sequentially updated) expected pose.

This work extends that of [Platt et al., 2001], which offers a way to avoid the complexity of planning in a high dimensional belief space. It does this in two ways, i) by approximating the informational value of actions from a low-dimensional subspace of the belief state; and ii) by embedding that informational value into the physical space. This enables standard motion planning techniques to trade off directly between information gain and achievement of the goal pose for the manipulator.

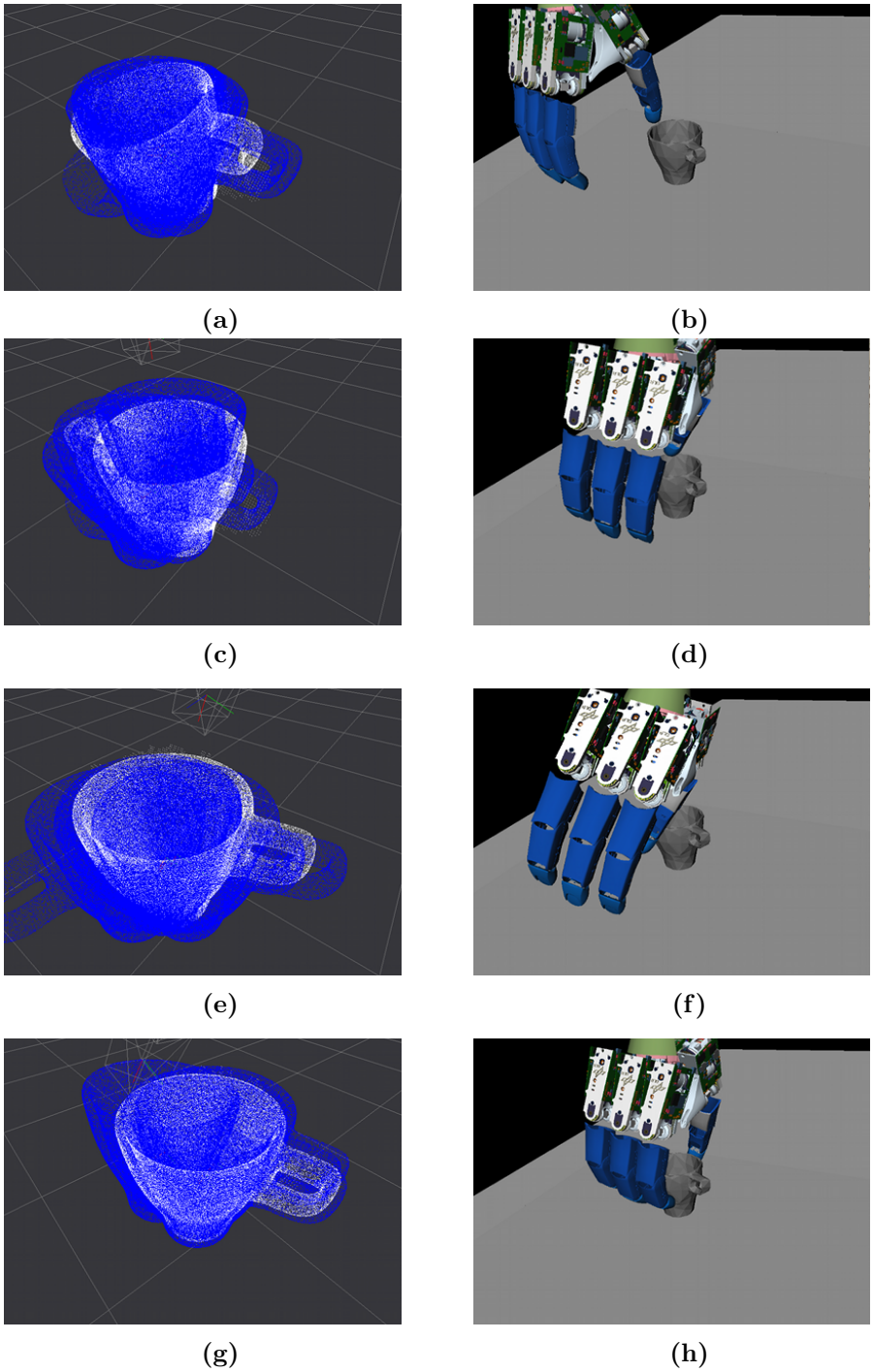


Figure 6.4: The belief states shown are the low dimensional belief states sub-sampled from the corresponding high dimensional belief states. Top row: Initial belief state (a), first contact (b). Second row: updated belief state from first contact (c), second contact (d). Third row: updated belief after second contact (e), third contact (f). Bottom row: updated belief after third contact (g), executed reach-to-grasp pose (h).

Chapter 7

Path planning for informational effects in the real world

From Chap. 6, we have seen that sequential re-planning can achieve better quality grasps than a single attempt in the presence of object pose uncertainty. We have also seen that trajectories designed to maximise tactile information gain achieve successful grasps with fewer iterations.

However, in the previous chapter, several assumptions were made which might be problematic for real world robot deployments. First, a complete model for the object to be grasped was available, in the sense of a dense point cloud or mesh model, as well as a pre-computed grasp for the object's model, therefore only uncertainty in the object pose remained. Second, the algorithms all assumed that noisy visual sensors were available to localise the target object, but relied on perfect tactile sensing abilities. Third, a bounding box was constructed around the object to be grasped. This allowed planning of collision-free reach-to-grasp trajectories only for convex objects. For example, a "rim" grasp on a mug would not be possible to plan because it requires placing at least one

finger inside the bounding box. In addition, each re-planning iteration was treated as independent and thus the robot's manipulator was withdrawn to a safe position after a failed attempt to grasp. Chapter 6 demonstrates the validity of the algorithms mainly in simulation and, only as an initial attempt at proof of concept, the algorithms were implemented and tested on the Justin robot using an object rigidly fixed to the table (see Fig 6.2).

These assumptions all prevent the core methods from working on a real robot. In this chapter we discuss how the previous algorithms can be extended and enhanced, to enable the relaxation of each of these assumptions, bringing us to a first convincing demonstration of this approach on a real robot. The resulting algorithms can also:

- Compute the target grasp on-the-fly, by incorporating the grasp planning method of [Kopicki et al., 2014].
- Interpret the noisy contact sensors of a real robot hand (using improved Bayes filtering).
- Re-plan trajectories without requiring withdrawal of the manipulator to a safe pose, but where the manipulator can remain in contact with the object.
- Planning dexterous grasping trajectories for non-convex objects.

This work is demonstrated in trials in simulation and on Boris, a half-humanoid robot platform. Empirical results confirm that sequential re-planning achieves a greater success rate than single grasp attempts, and trajectories that maximise information gain require fewer re-planning iterations than conventional planning methods before a grasp is achieved.

7.1 Organisation

This chapter proceeds as follows. In Sec. 7.2, we discuss the limitations of the set of algorithms presented in the previous chapter due to several assumptions being made. We also discuss how to relax these assumptions in order to validate the sequential re-planning approach on a real robot.

Section 7.3 presents the technical contributions of this chapter. We discuss a new method to explicitly address the multi-modal nature of the belief state for the problem of robot grasping. In addition, the section presents a set of engineering or algorithmic solutions which enabled us to implement the sequential re-planning approach for a real robot platform.

Section 7.4 presents empirical results in a real scenario in which a dexterous grasping trajectory has to be planned under non-Gaussian object-pose uncertainty in 6D with shape incompleteness. In addition, this approach is also demonstrated in a virtual scenario. Three strategies are presented.

Section 7.6 summarises the contributions of this chapter.

7.2 Introduction

In Chap. 2, we saw that the grasp planning problem is composed of four sub-problems: state estimation, grasp synthesis, grasp planning and control. As seen in Sec. 2.5, a typical approach is to represent the belief state using prior distributions (usually a Gaussian), select a more robust grasp in the face of uncertainty (pose uncertainty or shape incompleteness) and finally to use tactile feedback to adjust the grasping trajectory, see e.g. [Nikandrova et al., 2013]. The reach-to-grasp trajectory is typically computed using some conventional sampling-based techniques which minimise the cost, in Eu-

Table 7.1: ReGrasp & ReGrasp+IG vs Mycroft & IR3ne at a glance

	ReGrasp & ReGrasp+IG	Mycroft & IR3ne
Target grasp	Pre-computed	Computed on-the-fly
Pose estimation	Mean-shift algorithm	Mean-shift + clustering algorithm
Re-planning	From a safe robot pose	From current pose of the robot
Contact sensing	Assumed to be perfect	Gaussian filter to remove noise

clidean space, to transfer the robot's end effector to the selected grasp configuration. Comparatively little work has explored the more complex problem of reasoning about uncertainty while planning a dexterous reach-to-grasp trajectory. This is mainly due to the high dimensionality of the configuration space of a dexterous manipulator, and the complexity of determining the effects of an action on the object we wish to grasp. This chapter presents a new set of sequential re-planning algorithms that enable a robot to autonomously operate under object-pose uncertainty in a real scenario.

Nevertheless, the innovative approach I chose in this thesis comes at a cost. First, even in this chapter, it is assumed that a reference model for the object is available. Nevertheless, completeness in this model is not necessary for this set of algorithms to work. Incomplete point clouds can be used as models. Second, it is assumed that a separate sequence of views, given by a depth camera, give an incomplete point cloud which can be aligned to this model. The shape incompleteness of both the model and the observation of the object results in the object-pose uncertainty being encoded in a belief density for the pose. Given this, the approach presented here selects an active information gathering reach-to-grasp trajectory with respect to the current belief state.

Note that for the new implementation of the sequential re-planning algorithms, no further knowledge on the object we wish to manipulate is needed. Regarding the generation of the target grasp configuration, the system is capable of generating this for a novel object by using the techniques described in [Kopicki et al., 2014]. However, the choice of using a point cloud model of the object has other implications. In the literature, there are no fast

collision detection procedures for non-convex objects represented as point clouds. As a contribution to this thesis I have therefore developed an efficient KD-tree based collision detection algorithm that allows us to efficiently plan dexterous grasping trajectories in such a situation (Sec. 7.3.5). This is useful for real grasping. Point clouds have the appealing property that they can be built on-line and incrementally[1], and that they do not have to be complete. In some ways these three properties make point clouds a more desirable representation for manipulation than surface-based, meshes or volumetric representations, such as representation of shapes.

In Chap. 6, the IG planner works with a cost function that allows deviations from the shortest path. These encourage gathering tactile observations that will reduce pose uncertainty in the object location. Section 6.4.2 showed how to reason about the informational effects of an action by simply encoding the expected informational value into a cost function. Such a cost function can be minimised using any motion planning technique (here a PRM), such as the ones described in Chap 3. As seen at the end of the previous chapter, a drawback of encoding the information value into the cost function is the extra computation during the query phase of the PRM algorithm in order to compute the likelihood of reading a contact. This requires finding the closest surface to the finger tips for each hypothesis every time a node of the PRM is in proximity to the uncertain region. The new set of algorithms in this chapter also uses the procedure described in Sec. 7.3.5 to compute the expected (tactile) observations in the planning process (Sec. 6.4.1), erasing the computational discrepancy between the two sequential re-planning algorithms.

The rest of this chapter presents the new features implemented in the sequential re-planning algorithms in more detail.

[1]Registration procedures allow us to merge several point clouds, collected from different view points, into one.

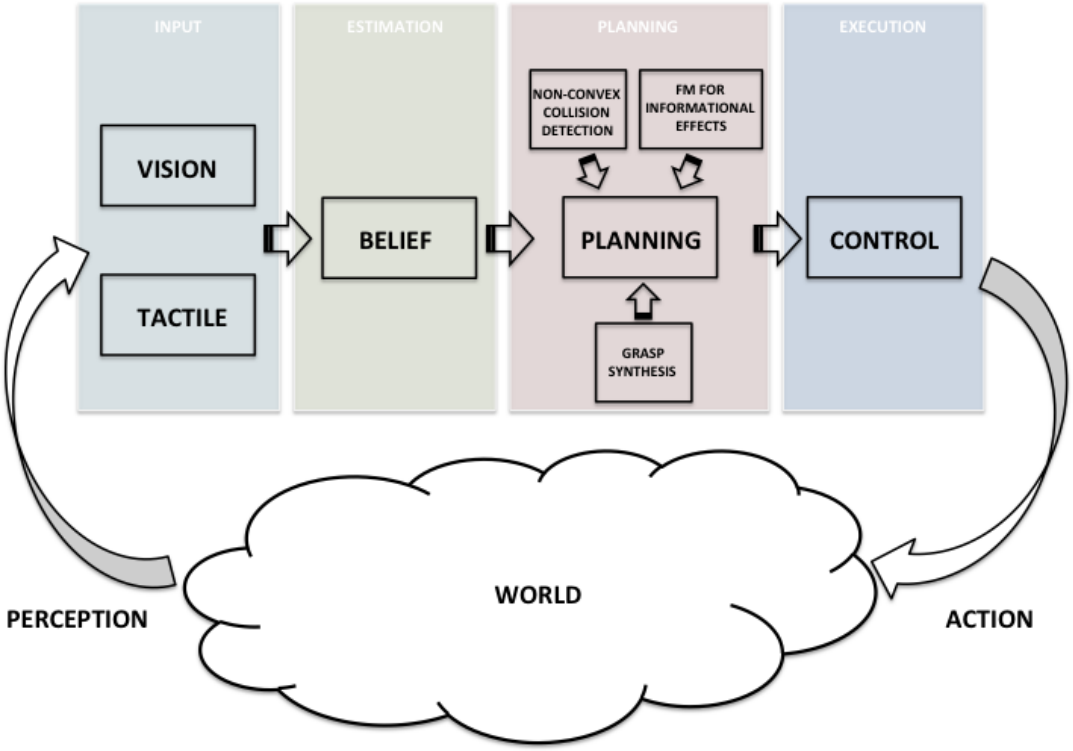


Figure 7.1: The architecture of the system is composed of 4 components: sensory input, pose estimation, motion planning and active control. The arrows show the flow of the system and its interaction with the external world. The system presented in this chapter involve 3 different modules in the motion planning phase: i) collision detection procedure for non-convex objects represented as a point cloud, ii) a forward model (FM) is used to maximise the chance of gathering tactile observations that will reduce pose uncertainty, and iii) a grasp synthesis procedure to compute grasps on point clouds.

7.3 Technical contributions

As explained, the set of algorithms presented in this chapter has a similar formulation to those in Chap. 6. However, there are several key differences to be discussed. This section describes each of them separately.

7.3.1 Mean pose estimate

Section 6.3.2 presented this as the mean of the set of particles in the belief filter. In this chapter a new state estimator is proposed. In chapter 6, the mean was estimated using a mean-shift algorithm on the particle set. This simple estimate of the mean does not work well with multi-modal beliefs. Therefore, in this chapter, I combine the mean-shift algorithm with a hierarchical clustering algorithm to compute the mean of the most promising cluster within a possible multi-modal belief.

This procedure uses the mean-shift algorithm to generate a set of cluster centres from the particle set. Each associated with: i) a score which identifies the "goodness" of the estimate in terms of how well the estimate represents the entire set of particles, and ii) the number of particles that have contributed to construct the estimate. Let c_i be the centre particle of a cluster, C_i, of the entire set of particles $[p_1, \ldots, p_K]$. For each particle the sampling-based model-fitting procedure described in Sec. 6.3.2 computes an importance sampling weight w_k, which expresses the likelihood of best aligning the (dense) model point cloud (assumed to be available in the system), to the (partial) query point cloud. The importance sampling weight can be considered as a measure of how good the pose p_k describes the data, and then the score for the cluster C_i is computed as

$$s_{C_i} = \sum_{k \in [1,\ldots,K]} w_k e^{\|c_i - p_k\|_Q}$$

where $\|a\|_Q$ is defined as $a^T Q a$ and Q is the covariance matrix of the set of particles $[p_1, \ldots, p_K]$. Once a fixed number of clusters C_i is generated, the procedure agglomerates the most similar clusters, if any, and computes the best estimates as the mean of the most promising cluster C_{max}, which is defined as:

$$C_{max} = \arg\max_{C_i} \frac{|C_i|}{\sum_j |C_j|} s_{C_i}$$

where s_{C_i} is the score associated with the cluster C_i, and $|C_i|$ is the cardinality of the cluster C_i in terms of number of members.

7.3.2 Grasp synthesis

Section 2.4.2 discussed how to learn grasp types that can be generalised to the shape of novel objects and discussed the work of [Kopicki et al., 2014; 2015], which proposes an efficient method to learn dexterous grasp types (e.g. pinch, rim) from a single example that is able to generalise within and across object categories, and with full or partial shape information. As a minor contribution to this thesis, I have integrated this method to enable the system to generate grasps on-the-fly. A user of the system can select the grasp type which will be automatically generated on the novel object's shape.

7.3.3 Re-planning

As seen in Chap. 6, the sequential re-planning algorithm planned trajectories assuming only the maximum likelihood observations given the current belief state. Therefore we need to rely on sensory feedback during the execution of the planned trajectory in order to detect whether or not unexpected observations occur. This triggers a belief update, using the observation gathered at execution-time, and consequently a re-planning phase.

As described in Chap. 6, and in [Zito et al., 2012a; 2013b], after a contact the manipulator was moved back to a safe configuration (e.g. outside the uncertain region) prior to each new reach-to-grasp trajectory being planned. This approach was necessary for technical reasons due to the robotic platform in use as it was tested on DLR's Rollin Justin robot. The only way to communicate with this platform is via its own controller. This controller accepts trajectories with extra parameters to set, for example, compli-

ance or activating/deactivating a joint's torque thresholds (or guards). However these parameters cannot be changed on-line during the execution of the trajectory and the controller forbids any movement once an active guard has been triggered. It is possible however to send a trajectory with disabled guards. Therefore the only feasible work around, after making a contact, was to withdraw the robot to a safe configuration with no active guard before the next reach-to-grasp attempt. This, however, is inefficient and fails to exploit the existing contact in completing the grasp.

To overcome these limitations the modified method, proposed in this chapter, has been implemented on a robotic platform called Boris. Boris' controller enables us to modify settings (compliance and thresholds) on-the-fly at execution time. This allows us to make a contact with an object, stop the robot, and then re-plan from the same configuration, while maintaining contact with the object.

In the experiments presented here, the algorithm uses torque sensors, based on current draw, at each joint of the robot's hand to detect whether or not a link of the hand is in contact with the environment. As a minor contribution to this thesis, a contact detection module has been developed to remove the noise from the sensors, which we will discuss briefly now.

7.3.4 Detecting contacts

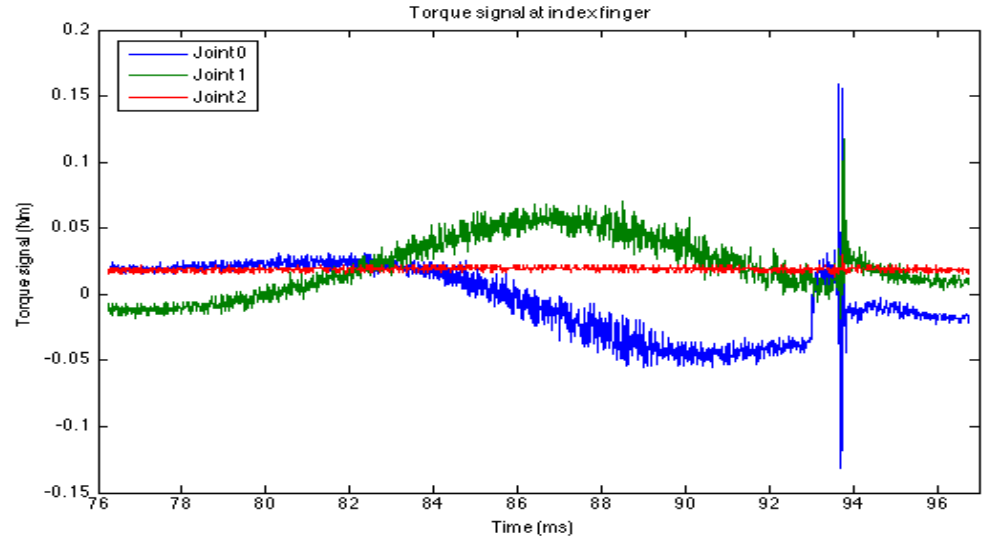
As mentioned, the system developed in this thesis relies on the torque sensors based on the current draw at each joint of the DLR Hit Hand II. Figure 7.2(a) shows the torque signal received from the robot hand during a reach-to-grasp trajectory. Joint 0 (blue) is responsible for the abduction movement, while Joint 1 (green) and 2 (red) are responsible for flexion movements. The fourth joint (Joint 3) of the DLR HIT Hand II is mechanically coupled with Joint 2, so it is not shown in the figure.

Algorithm 5 LOWPASS_FILTER

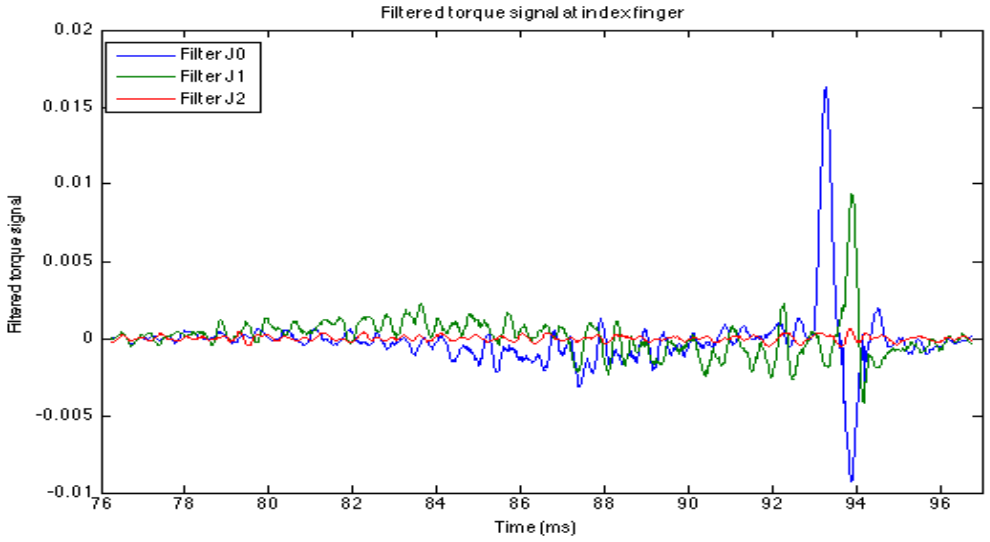
Input: data, mask_size, σ
Output: L
  x ← [-mask_size:.1:mask_size]
  mask ← diff(normpdf(x,0,σ))
  L ← conv(data, mask)

When the robot is commanded to move the inertias of each finger link cause significant torque signals to be sensed, even without any contact with external objects. Therefore an important practical problem is how to remove the portion of the torque signal that is due to the robot's own motion, in order to detect torque changes due to external contacts. To do so, a Gaussian low-pass filter with a fixed window size is used to filter out the noise. Alg. 5 shows the pseudo code for a low-pass filter. The functions diff($\cdot$), normpdf($\cdot$) and conv($\cdot$) are considered as in the MATLAB API, where diff(X) calculates differences between adjacent elements of X along the first array dimension whose size does not equal 1; normpdf(X, μ, σ) computes the pdf at each of the values in X using the normal distribution with mean μ and standard deviation σ; and conv(u, v) returns the convolution of vectors u and v. In this implementation $mask\_size$ = 2 and σ = 1.0, while *data* is the vector of torques for a particular joint. The length of *data* is equal to the selected window size. In this implementation the window size is equal to 40, which means *data* contains the latest 40 readings from the joint's torque.

Figure 7.2(b) shows the corresponding filtered signals along the trajectory. Ideally, we would like to have a zero signal until the finger makes contact with the object. However, this is not possible to achieve due to the activity of the joints' actuators, the acceleration of the hand along the trajectory and the changes in the gravity vector according to the changes in the orientation of the hand. Nevertheless, the system allows us to define thresholds to further filter the signals. Via empirical experiments on Boris we determined that a threshold of 0.05 gives an appropriate trade-off between false positives and false negatives.



(a)



(b)

Figure 7.2: The images show the real torque signals (a) and the corresponding Gaussian low-pass filter signal (b) for each joint of the index finger of the DLR Hit Hand II during a reach-to-grasp trajectory. Joint 0 (blue) is responsible for the abduction movement, while Joint 1 (green) and 2 (red) are responsible for flexion movement. The fourth joint (Joint 3) of the DLR Hit Hand II is coupled with Joint 2, so it is not shown. The torque signal is plotted over time. At approximately 93ms a contact with the target object is made, as shown by the increasing of the torque signals.

7.3.5 Planning a dexterous grasping trajectory for non-convex objects

The implementation of this planner uses a modified version of Probabilistic Roadmap (PRM) planning, [Kavraki & Svestka, 1996], to plan trajectories. Crucially, sampling-based approaches like PRMs rely on the ability of rejecting points that are in collision. Testing collision between simple geometrical primitives, such as planes, spheres or cubes, is substantially faster than between non-convex polyhedra. On the other hand, if an object to be grasped is wrapped in a simple primitive bounding box, to achieve computational efficiency, many grasps would simply be impossible to achieve. The left image in Fig 7.3 shows the case of a grasp over the rim of a jug. This grasp requires the thumb to penetrate inside the convex hull of the object. Thus, even if this configuration does not produce any true collisions, it would be rejected by such "naive" collision detection.

Here, a fast and efficient collision detection module to cope with objects represented by a point cloud is proposed. Whilst the robot's rigid links are represented by an open-chain of convex polyhedra, the object to be grasped is represented by a 2-level structure. The first level of this structure wraps the point cloud in a bounding box. This level can efficiently avoid checking collisions between the robot's links and the object to be grasped when they are far apart. The second level contains the point cloud organised in a KD-Tree. The KD-Tree implementation is based on the FLANN library [Muja & Lowe, 2014]. An example of the 2-level collision detection is shown in Fig. 7.4.

The planning process uses a collision detection in two cases: i) to reject PRM nodes in collision with an object and ii) to compute the information value for each edge of the PRM. In the former case, the planning process only targets the object as if it were in its expected location (the mean pose of the density function), therefore only the mean pose is used for collision detection. In contrast, the latter case requires us to compute the information value for all the sub-sampled hypotheses. In both cases, when at least

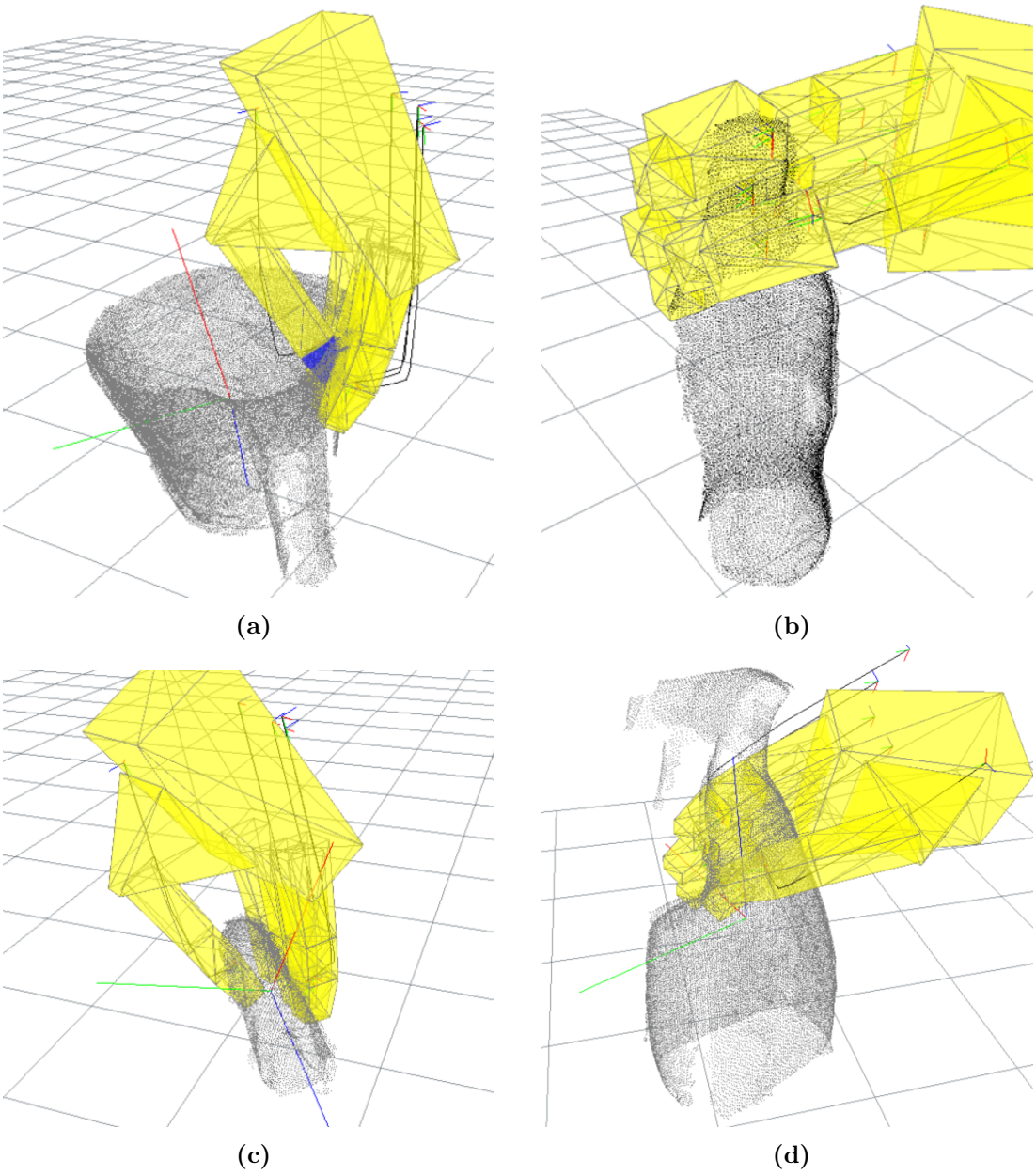


Figure 7.3: Models of the objects and their associated grasps. The image (a) shows a rim grasp on a jug. In this case, the target grasp requires that the thumb be placed on the internal surface of the object, thereby penetrating inside the convex hull. The image (b) shows a top grasp on a bottle of coke. The images (c) and (d) show respectively a rim grasp on a stapler and a Mr Muscle spray bottle. The grasp configurations are computed using the method described in [Kopicki et al., 2014].

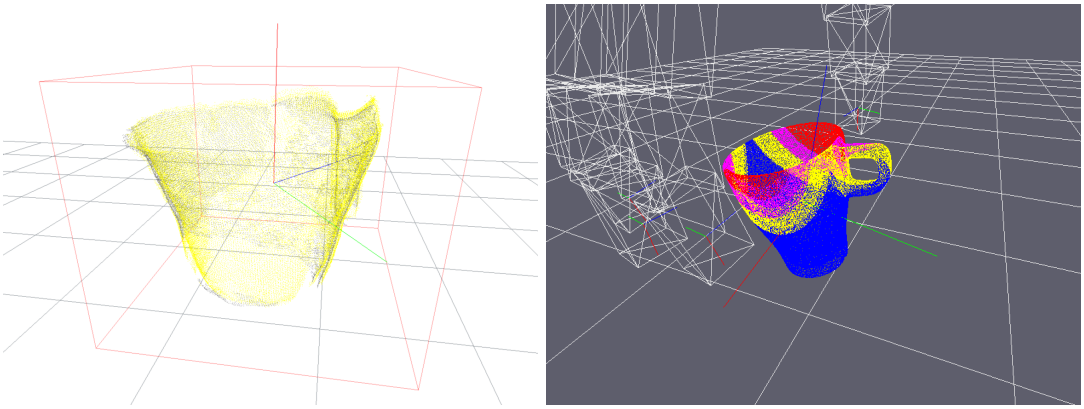


Figure 7.4: The left image shows the target object to be grasped. In this case the object is a jug. The grey point cloud represents the ground truth pose of the object, as it was acquired by a noiseless input source. The yellow point cloud identifies the best pose estimate. The yellow point cloud is organised as a KD-Tree for faster collision detections. The red box represents the bounding box, which prevents unnecessary collision checking between the object and robot's links when they are far apart. The right image shows a point cloud for a mug and the robot hand's configuration. Each point is coloured with respect to the relative distance to the closest finger's link of the robot's hand: red (closest) to blue (furthest).

one bounding box of the robot collides with the bounding box of the object, the second level of the collision detection module is called. By querying the KD-Tree it is possible to retrieve the closest points on the surface of the object to the robot's links, see Fig. 7.4 (right). Each point is then checked to determine whether it collides or not, or to estimate the likelihood of observing a contact.

Algorithm 7 shows the core of the proposed collision detection procedure. This procedure computes a *penetration* value for a particular point p in the point cloud and a set of bounds for the robot hand. The penetration value is an approximation of the distance between p and the closest surface of the robot hand, and it is positive if the point sits inside the bounds, or negative otherwise. Algorithm 6 shows the collision detection procedure used to reject a robot pose in collision. In this case, the procedure rejects a robot pose if the associated penetration value is greater than or equal to zero. It is also possible to treat a point p as a sphere with centre p and radius ρ and this allows us to plan more conservative trajectories.

Let b_i be the boundary of the i–th link in the robot hand, represented as a convex polyhedron. Figure 7.5 shows the boundary b_i as a simple 6 faced polyhedron. Then b_i is composed of a set of triangles $T_i = [t_1, \ldots, t_M]$ and a reference frame $O_i \in SE(3)$. Each triangle $t_j \in T_i$ is composed of three vertices v_{j1}, v_{j2}, v_{j3} and it is possible to compute the normalised normal vector n_j as

$$n = (v_2 - v_1)(v_3 - v_1)$$

$$n_j = \frac{n}{\|n\|}$$

Note that either n_j or $-n_j$ would be a normal for the triangle t_j. The procedure chooses n_j with the direction that points outside the polyhedron, as shown in Fig. 7.5. Note that since all the normals are computed to point outside the boundary, for a point p to be in collision it must satisfy the following condition:

$$\|v_{j1}\|_{O_i} - n_j^T p > 0, \qquad \forall j \in [1, \ldots, M]$$

where $\|v_{j1}\|_{O_i}$ is the distance of the triangle t_j from the reference frame O_i, approximated as the distance to its first vertex v_{j1}, and $n_j^T p$ is the dot product between the normalised normal n_j and the point p. In classical geometry, the distance between a triangle and a point is computed as the length of the projection of the point p onto the triangle or, if the projection does not sit on the triangle's surface, distance to the point's projection onto the closest edge of the triangle. In order to speed up the computation, the implementation I propose computes the penetration value for each triangle as the distance between the point p and the first vertex, v_1, of the closest triangle. If the point sits outside the boundary b_j, the penetration value is the negative of the computed distance. If the triangles t_j are small, this is a good approximation.

Note that the lack of complete information about the object's shape may affect collision

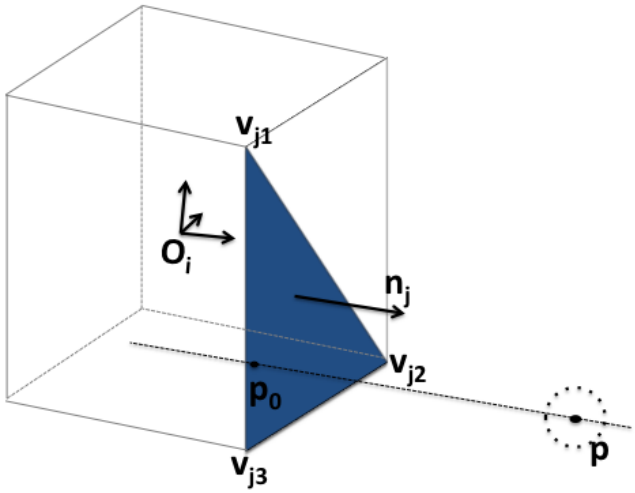


Figure 7.5: The image shows a mesh triangle, t_j, of vertices v_{j1}, v_{j2}, v_{j3} and the normalised normal vector n_j, the reference frame of the mesh O_j and the point p with its projection on the triangle's surface. The projection is the point p_0 on the triangle surface which intersects the line passing through the point p with the same direction of n_j. The dotted line around the point p represents the spherical bounding box used for planning more conservative trajectories.

Algorithm 6 CHECK_COLLISION

Input: robot_pose, hand_bounds, kdtree, neighbours, ρ
  closest_points ← KNN_SEARCH(kdtree, robot_pose, neighbours)
  for all $b_i \in$ hand_bounds **do**
    $d \leftarrow$ GET_PENETRATION(b_i, closest_points)
    if $d > -\rho$ **then**
      return true
    end if
  end for
  return false

detection, leading to a reach-to-grasp trajectory which passes through the object. In a real scenario however, the tactile observation that will be generated will cause the robot to stop and the current belief state to update. One limitation of the current

Algorithm 7 GET_PENETRATION

Input: b_i, closest_points
  distance $\leftarrow +\infty$
  for all $p \in$ closest_points **do**
    $d \leftarrow$ GET_PENETRATION_PER_BOUND(b_i, p)
    if distance $> d$ **then**
      distance $\leftarrow d$
    end if
  end for
  return distance

Algorithm 8 GET_PENETRATION_PER_BOUND

Input: b_i, p
  distance $\leftarrow +\infty$
  direction $\leftarrow +1$
  for all $t_j \in b_i$ **do**
    $d_1 \leftarrow \|tj.v_{j1} - p\|$
    if distance $> d_1$ **then**
      distance $\leftarrow d_1$
    end if
    if direction > 0 **then**
      $d_2 \leftarrow \|t_j.v_{j1}\|_{O_i} - t_j.n_j^T p$
      if $d_2 < 0$ **then**
        direction $\leftarrow -1$
      end if
    end if
  end for
  return direction·distance

implementation, is that this approach does not deduce the relative likelihood that the unexpected observation is given by a mis-estimation of the object pose versus lack of shape information; it simply updates the belief density over the object-pose. In future work, the aim would be to treat this problem as a SLAM problem, which will enable us to compensate for initially incomplete shape information in the object model.

7.3.6 Terminal conditions

The sequential re-planning algorithm terminates its execution when no unexpected contacts occur and the target grasp is achieved. Since we do not have mesh models of the object to be grasped we cannot rely on grasp quality measures (discussed in Sec. 2.4.1) to signal successful termination of the algorithm. Nonetheless, in simulation it is possible to measure the displacement error between the grasp configuration for the final object-pose estimate and the ground truth. A user-defined threshold of tolerance is used to identify whether the grasp has succeeded. On the real robot, the success of the grasp is evaluated by lifting the object. If the robot can hold the object, the grasp is considered successful.

7.4 Results

This section presents the experimental results used to evaluate the new set of algorithms presented in this chapter. As in Chap. 6, three strategies for planning dexterous reach-to-grasp trajectories are evaluated both on the real robot platform Boris and in a simulated environment:

- ELEMENTARY: an open-loop trajectory towards the expected object pose without re-planning.
- MYCROFT: a sequential re-planning algorithm without information gathering.
- IR3ne: a sequential re-planning algorithm with information gathering.

In this evaluation the aim is to show that sequential re-planning is capable of achieving higher grasp success rates than single grasp attempts in the presence of non-Gaussian object-pose uncertainty in 6 dimensions. It also shows, as in the previous chapter, that planning trajectories that maximise information gain result in fewer re-planning

iterations to achieve a grasp.

First, Sec. 7.4.1 presents empirical results collected on 20 trials for a single object (a jug) using a real robot, in which the ability of these three different strategies to achieve a grasp configuration is tested, as well as the benefits of using a mean-shift algorithm with hierarchical clustering to compute the estimate of the object's pose.

In addition, Sec. 7.4.2 presents the results collected on 120 trials in simulation for four objects: a jug, a bottle of coke, a stapler and a Mr Muscle spray bottle. In these experiments a mean-shift algorithm is still used to compute the pose estimate, and the aim is to evaluate the ability of the three strategies to achieve a grasp configuration.

7.4.1 Experiments on the robot Boris

These experiments are composed of 2 runs of 20 trials each (10 trials using MYCROFT and 10 using IR3ne). The results for the ELEMENTARY strategy are extrapolated from the results collected for MYCROFT, in the sense that the two approaches construct a reach-to-grasp trajectory minimising the same cost function, therefore if MYCROFT achieves (or fails to achieve) a grasp at the first iteration, the ELEMENTARY also would have succeeded (failed) as well.

For both set of trials only one object has been used: the jug shown in Fig. 7.4. A dense point cloud, representing the object model, is assumed to be available at the beginning of each run. This object model is acquired by scanning the operational workspace (Sec. 3.2.5) with a depth camera from 6 different views. These views are merged to generate a single point cloud of the object model. This pre-processing aligns the single view point clouds, registering and eliminating outlier points.

I argue that in order to evaluate the ability of planning more robust reach-to-grasp trajectories in the face of pose uncertainty, a critical parameter is not the shape of the

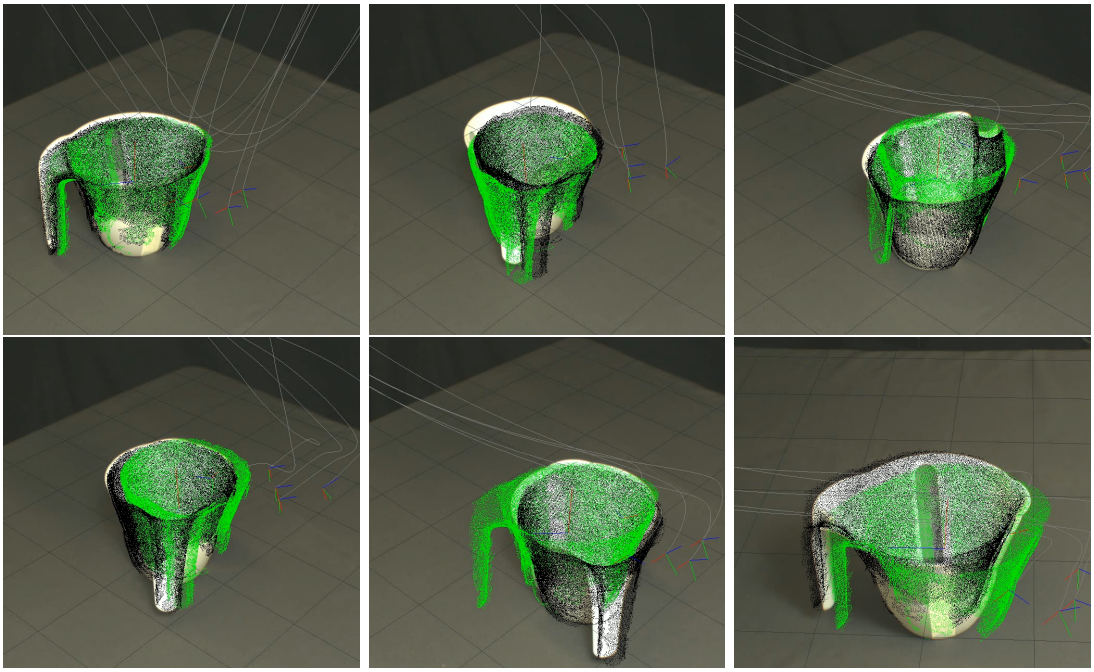


Figure 7.6: Initial pose uncertainty. The sequence of images shows some examples of initial pose estimation during the experiments on Boris. The simulated point cloud overlay the real plastic jug to show the misalignment between the real pose of the object, the ground truth (black point cloud) used to evaluate the simulated results and the initial pose estimate (green point cloud), to which the algorithm attempt the grasp. The lines show the planned trajectory for each fingers.

object but the initial estimation error w.r.t. the ground truth. Thus, at each trial, the target object is placed on a table in front of the robot, within the robot's operational workspace, in a different configuration. In this case, the query point cloud is acquired by scanning the operational workspace from only 3 different views, which produces a mismatch between the estimated pose and the ground truth. Once these views are aligned and registered, Boris estimates the belief state with a set of particles. The first run used a mean-shift algorithm combined with the hierarchical clustering algorithm, as presented in Sec. 7.3.1, to estimate the object's mean pose. The second run used only the single mean-shift algorithm, as described in Sec. 6.3.2. Figure 7.6 shows some of the initial estimations from the first run of the experimental trials. For clarity, the black point cloud represents the ground truth computed with the same model-fitting algorithm

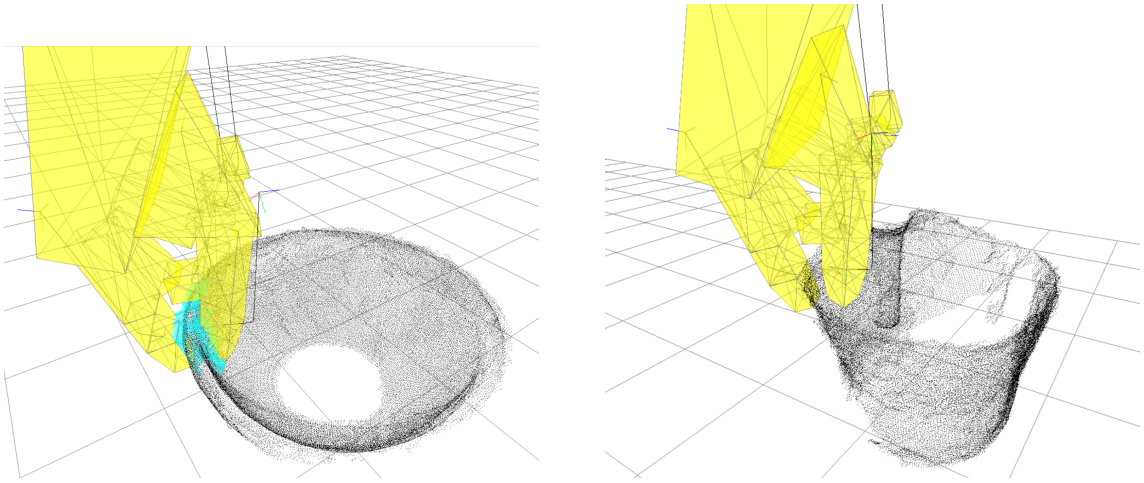


Figure 7.7: Pinch with support grasp learned on a bowl and transferred to a jug, using the method described in [Kopicki et al., 2014]. The image on the left shows the learned contact model (blue points) for all the fingers involved in the grasp. The right image shows a possible grasp adaptation on a jug.

used in Sec. 6.3.2, but sampling 500 times more features. The ground truth is also used in the experiments in a virtual scenario as explained in the next section .

In both runs, a target grasp for the jug is computed by adapting a pinch with support grasp learned on a bowl. The training and test grasps are shown in Fig. 7.7. Then a reach-to-grasp trajectory is computed and performed. A trial is considered successful if Boris can converge to the target grasp configuration and lift the object from the table surface.

Figure 7.8 summarises the empirical results collected. In order to test the ability of a sequential re-planning algorithm to converge to the "true" pose of the object, a ground truth pose of the object is calculated, at the beginning of each trial, by using the same model-fitting algorithm as used in Sec. 6.3.2, but sampling 500 times more features. The algorithms have no knowledge of the ground truth pose.

The results are organised in Fig. 7.8 as follows:

- The number of average planning iterations across all the trials for each run, for

both re-planning strategies: MYCROFT and IR3ne (Fig. 7.8 top left chart).

- The success rate across all the trials for each run, in the sense of their ability to converge to the planned grasp for all the three strategies (Fig. 7.8 top right chart).
- The average reduction in the positional error between the estimated location and the ground truth across all the trials for each run. The error is computed as the Euclidean distance in a 3D space for both strategies: MYCROFT (middle left chart) and IR3ne (Fig. 7.8 middle right chart).
- The average reduction in the rotational error between the estimated rotation and the ground truth across all the trials for each run. The error is computed as the distance in the quaternion space for both strategies: MYCROFT (bottom left chart) and IR3ne (Fig. 7.8 bottom right chart). The rotational error is computed for each orientation on the unit hypersphere in 4D quaternion space, and then displacement is measured as the length of the arc of the geodesic which connects these two points.

Section 7.5.1 discusses the presented results.

7.4.2 Experiments in a virtual environment

The experiments are composed of 120 trials per object in a virtual environment: a jug, a coke bottle, a stapler and a Mr Muscle spray bottle. Each trial has a different initial probability density over the object pose. We tested the ability of different strategies to achieve a grasp configuration. The algorithm has a model of the object to be grasped, in the form of a dense point cloud, computed by scanning the object with a depth camera from 7 different views. As before, these views are pre-processed to generate a single point cloud of the object model. The pre-processing aligns the single view point clouds, registering and eliminating outlier points. Figure 7.3 shows the pre-processed

point clouds of each simulated object, and the associated target grasp configurations. For these experiments the models are composed of 7 single-view point clouds, apart from the bottle of coke which is composed of only 5 views.

The algorithms have been tested under the hypothesis that, at each trial, the object is displaced to a different position - but still in the dexterous workspace of the robot - and the robot has to attempt a grasp even if the new point cloud is not as dense as the model point cloud. In simulation, I achieve this by applying a rigid body transformation to the single-view point cloud to move it to a different location within the dexterous workspace of the manipulator. Four different conditions have been tested. In each condition we randomly selected either 1, 3, 5 or 7 view point clouds, from the 7 single-views collected, in order to simulate real situations in which the robot's depth camera has been able to observe smaller or larger parts of the object. Once the subset of views has been selected and moved to the simulated object location, the state estimation procedure described in 6.3.2 is performed.

Figures 7.9, 7.10, 7.11, and 7.12 summarise the data collected in our experiments. For each condition mention above a model coverage is computed in terms of which percentage of the model surface was covered by the merged point clouds from the individual views. All the results are compared with respect to the model coverage in percentage terms. Again, a ground truth pose for the object is calculated at the beginning of each trial by using the same model-fitting algorithm used in Sec. 6.3.2, but sampling 500 times more features. The ground truth is also shown in Fig. 7.6 as a black point cloud. The algorithms have no knowledge of the ground truth pose, however in the virtual environment the ground truth is used to model the real object location, and is used to trigger simulated contacts with the robot hand. These simulated contacts cause the sequential re-planning algorithm to stop and update the belief state.

The results in Figures 7.9, 7.10, 7.11, and 7.12 are organised similarly to the experiments

on the real robot:

- The number of average planning iterations across all the trials for each condition (1, 3, 5, and 7 single-view query), for both the re-planning strategies: MYCROFT and IR3ne (top left chart).
- The success rate across all the trials for each condition, in the sense of their ability to converge to a grasp for all the three strategies (top right chart).
- The average reduction in the positional error between the estimated location and the ground truth across all the trials for each condition. The error is computed as the Euclidean distance in a 3D space for both strategies: MYCROFT (middle left chart) and IR3ne (middle right chart).
- The average reduction in the rotational error between the estimated rotation and the ground truth across all the trials for each condition. The error is computed as the distance in the quaternion space for both strategies: MYCROFT (bottom left chart) and IR3ne (bottom right chart). The rotational error is computed for each orientation on the unit hypersphere in 4D quaternion space, and then displacement is measured as the length of the arc of the geodesic which connects these two points.

Section 7.5.2 discusses the presented results and illustrates the behaviour of the sequential re-planning approach (IR3ne) generated by one simulated trial.

7.5 Discussion

In this section we discuss the results presented in the previous section.

7.5.1 Sequential re-planning in a real scenario

Figure 7.8 presents the collected results on the Boris robot platform, as described in Sec. 7.4.1. The results show the benefits of using the hierarchical clustering algorithm to explicitly address the multi-modal nature of the belief space. In particular, the bottom row shows that, in the case of the jug, this approach leads to a lower initial rotational uncertainty with respect to a single mean-shift approach. This is due to the fact that, if the handle of the jug is not visible from the query point cloud, the object looks almost symmetric, which results in a multi-modal rotational uncertainty. Figure 7.13 shows this case from one of the trials from the collected results. Nevertheless, IR3ne is capable of achieving a grasp in a single iteration, thanks to a more robust approaching trajectory with respect to the estimated uncertainty.

Figure 7.14 shows an ideal case where the initial pose estimation does not cover the ground truth, in the sense that the belief state has no hypotheses that can explain the real pose of the object. Hence a first contact with the object (shown in Fig 7.14(b)) is not sufficient to produce an accurate estimation after the belief update. This leads to a second failed attempt, but also to a contact which allows Boris to locate the object and grasp it at the third, and final, attempt. Notice that this example is not part of the experimental results, but it has been recorded during a demonstration.

These two sets of trials have shown the validity of the sequential re-planning approach on a real scenario, however there are several issues that have to be addressed. First, the need for a predictive model for predicting how the target object moves after a contact occurs. Although the contact detection module described in Sec. 7.3.4 reliably stops the

robot after a contact is made, light objects, such as the plastic jug used in these trials, may be perturbed by the contact. For example, often we observe that the jug is tilted by a finger. The belief update by itself cannot deal with these cases and the resulting mean estimations are usually incorrect. However, the choice of point cloud models negatively affects the ability of developing predictive models. Second, the lack of tactile sensors does not allow us to have a good estimate of which link of the finger experienced the contact. This results in an additional uncertainty in the belief update.

Future work aims to address such issues by extending the sequential re-planning approaches to the use of tactile sensors and simple heuristics for an object-independent predictive model. The latter should be based on simple geometric properties of non-penetration between objects and robot's fingers to constrain the belief update and the mean pose estimate.

7.5.2 Sequential re-planning in a virtual scenario

The results collected in the virtual scenario confirm the ability of the sequential re-planning approach to achieve a grasp more robustly than a single grasp attempt (ELEMENTARY strategy) for all the objects (Fig 7.3) and all the conditions (see Sec 7.4.2). In addition, Sec 7.4.2 has also shown that IR3ne is capable of achieving successful grasps with fewer iterations than MYCROFT.

To illustrate the behaviour of the re-planning system Fig. 7.15 shows a typical sequence generated by one simulation trial attempting to perform a rim grasp on the jug. In this case, the initial belief state over the object pose is quite narrow (Fig. 7.15(b)), however an error of a few millimetres in the pose estimate leads to a potentially dangerous contact with the object (Fig. 7.15(c)). The contact is used to update the belief and plan a new trajectory from the current configuration of the robot(Fig. 7.15(d)). The second trajectory successfully transfers the hand to the target grasp configuration

(Fig. 7.15(e)).

7.6 Conclusion

This chapter has shown the validity, via empirical results, of the sequential re-planning algorithms on a real scenario. In addition, this approach has been evaluated in a simulated environment on four different objects. Several innovations have been developed to extend the theoretical techniques of Chap. 6, to overcome difficult practical problems encountered when trying to implement simultaneous perception and grasping with a real robot manipulator with 21 DoF and non-Gaussian object pose uncertainty in 6D, as well as incomplete knowledge of the object shape.

The main contribution of this chapter is to describe, demonstrate and evaluate a novel approach for the problem of robot grasping, where a robot is capable of reasoning about object-pose uncertainty while planning a dexterous reach-to-grasp trajectory. This system shown in Fig 7.1 is able to plan grasp operations autonomously in unstructured environments, with no knowledge of the object we wish to manipulate, apart from a model point cloud.

As in Chap 6, this chapter has shown how sequential re-planning can achieve better quality grasps than single attempts to directly grasp an object at its expected pose, and that re-planning with trajectories designed to maximise tactile information gain, achieves successful grasps with fewer iterations than sequential attempts to move directly towards the object's expected pose.

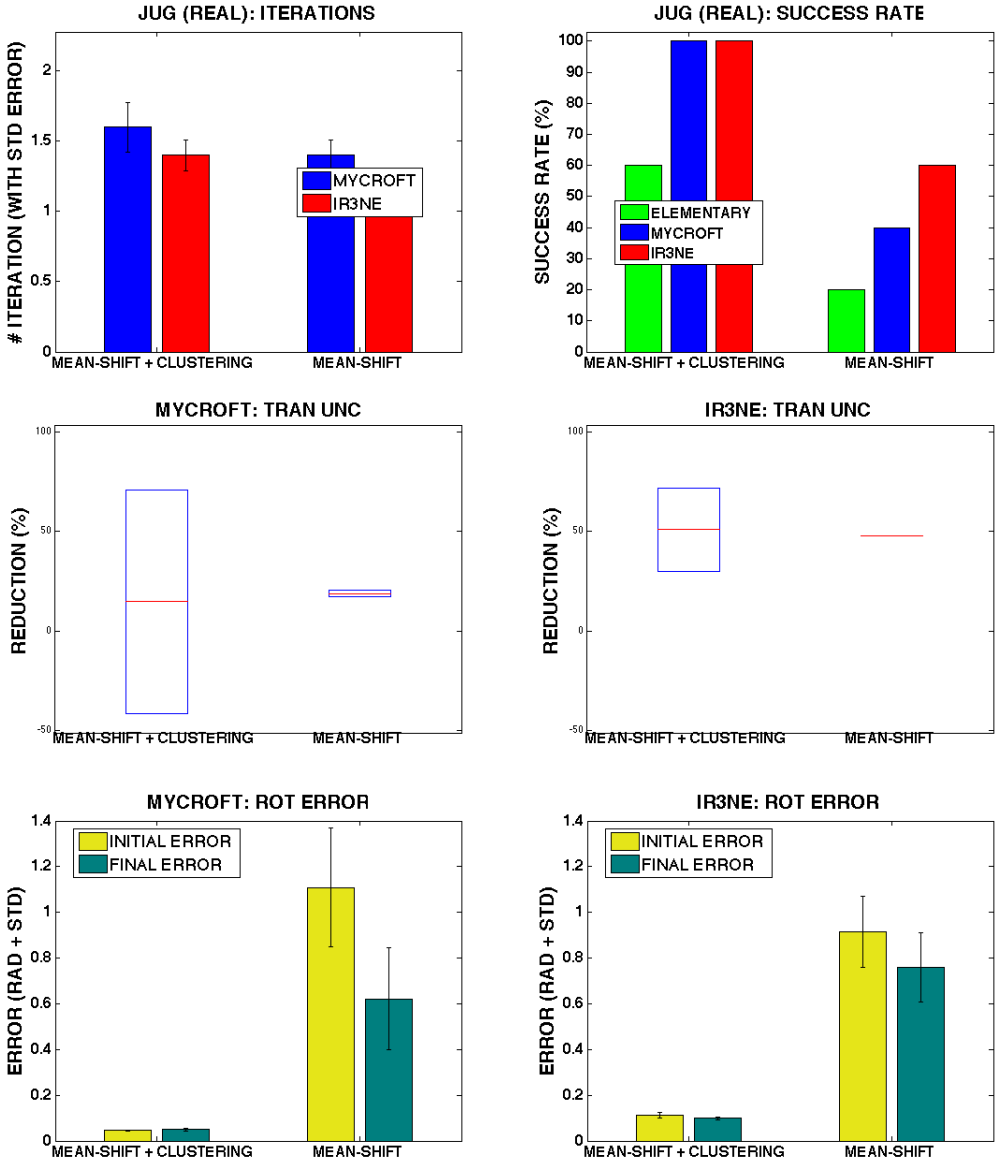


Figure 7.8: Empirical results on the Boris robot platform. The results refer to 2 sets of 10 trials on a jug. Three strategies were used: ELEMENTARY (green), MYCROFT (blue), IR3ne (red). Two conditions are presented. The first uses a mean-shift algorithm combined with a hierarchical clustering algorithm to estimate the object's pose. The second uses only a mean-shift algorithm. The top left chart presents the averaged number of iterations across the trials to reach a grasp. The top right chart shows the success rates across the trials. The middle row shows how the initial linear uncertainty (in percentile) is reduced by sequential re-planning until the algorithms converge to a grasp. The bottom row presents the reduction in the rotational uncertainty. Yellow bars represent the initial rotational error, while green bars are the final error. The rotational error is computed in quaternions where similar rotations yield an error value close to zero and rotations with 180 degrees difference yield an error value of 1.

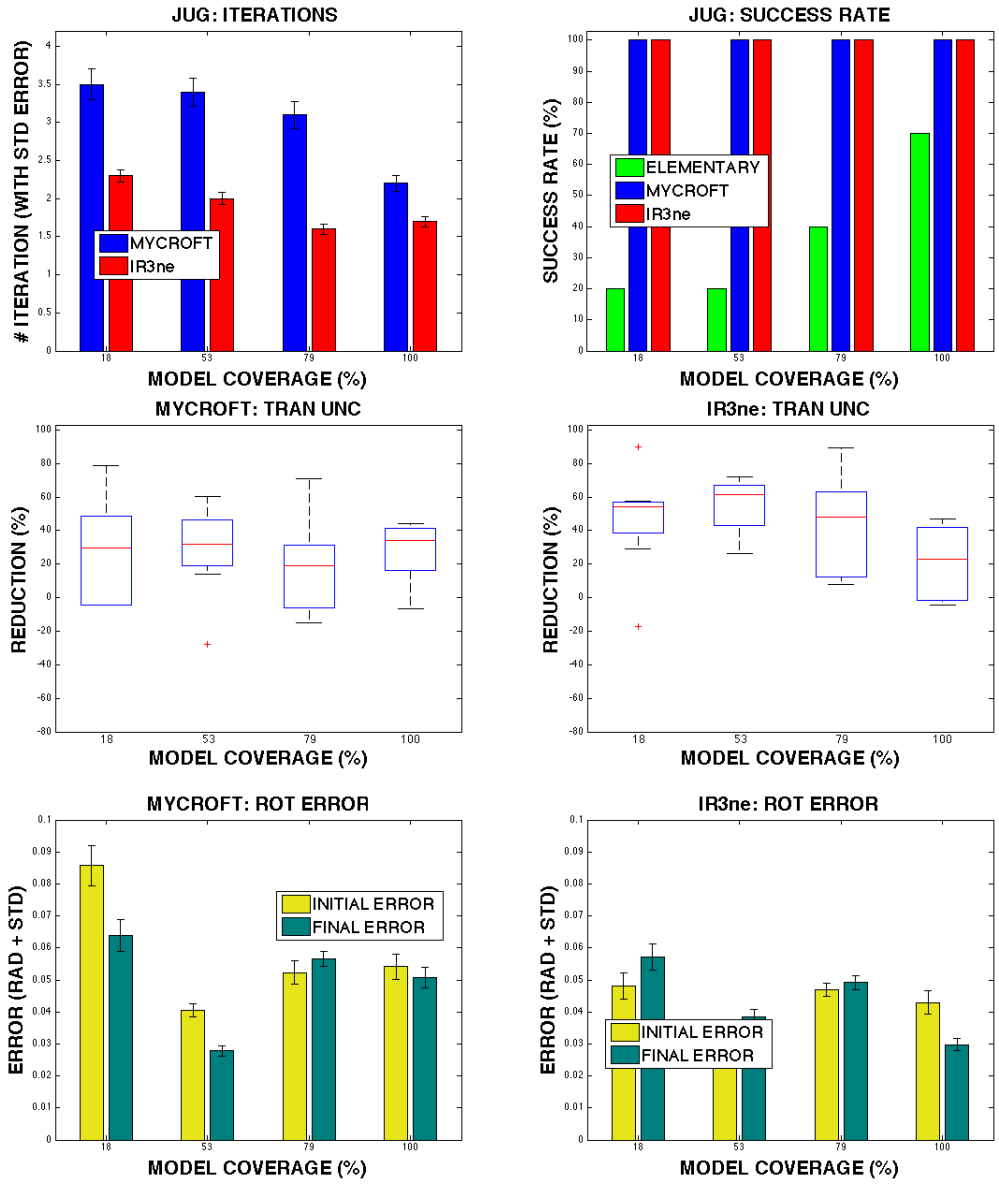


Figure 7.9: Simulated results for 120 trials on a jug. Three strategies were tested: ELEMENTARY (green), MYCROFT (blue), IR3ne (red). All the results are plotted against the initial percentage of model coverage for a total of four conditions. The top left chart presents the average number of iterations across the trials to reach a grasp. The top right chart shows the success rates across the trials. The middle row shows how the initial linear uncertainty (in percentile) is reduced by sequential re-planning until the algorithms converge to a grasp. The bottom row presents the reduction in the rotational uncertainty. Yellow bars represent the initial rotational error, while green bars are the final error. The rotational error is computed in quaternions.

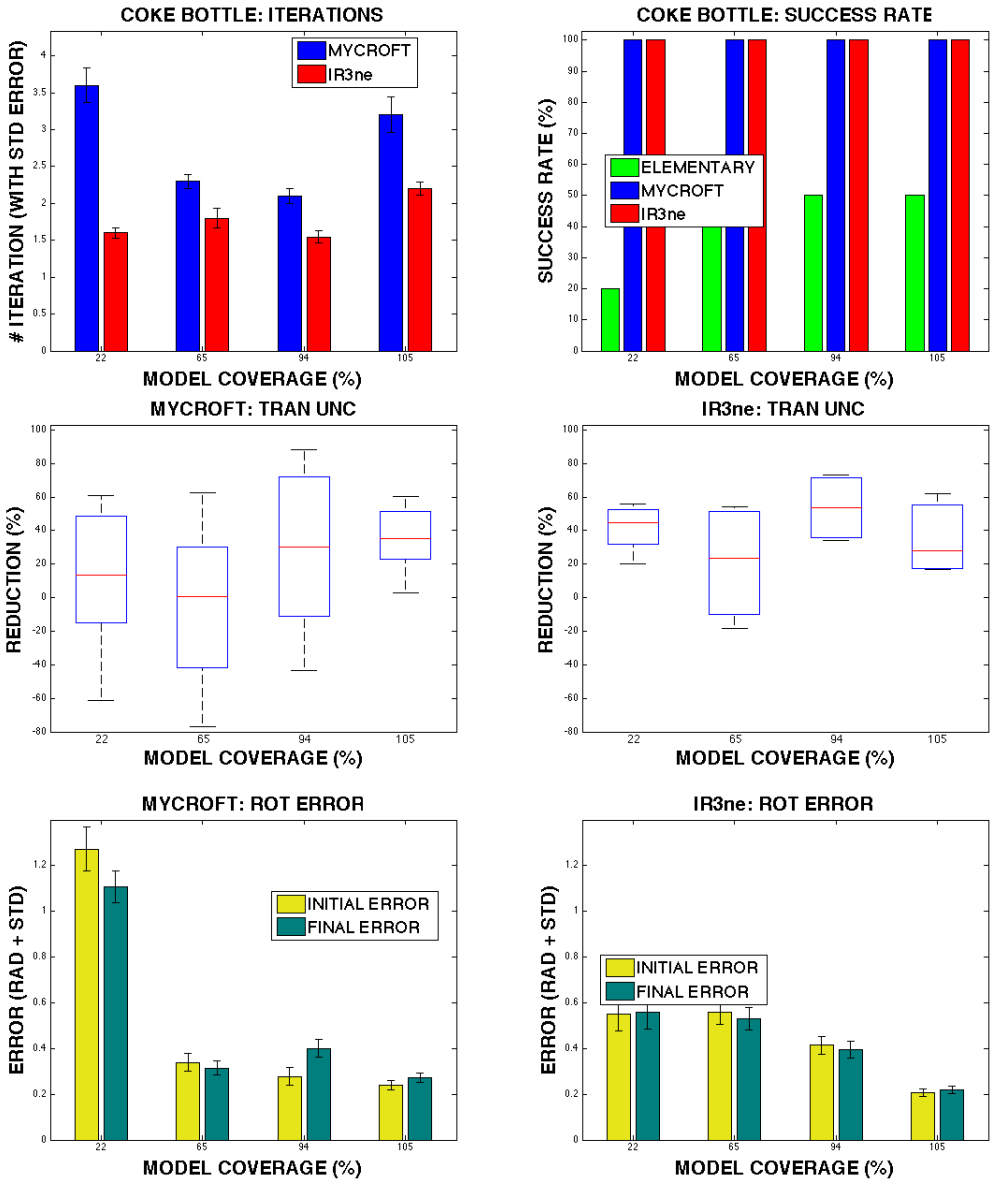


Figure 7.10: Simulated results for 120 trials on a coke bottle. Three strategies were tested: ELEMENTARY (green), MYCROFT (blue), IR3ne (red). All the results are plotted against the initial percentage of model coverage for a total of four conditions. The top left chart presents the average number of iterations across the trials to reach a grasp. The top right chart shows the success rates across the trials. The middle row shows the linear uncertainty reduction (in percentile), while the bottom row presents the reduction in the rotational uncertainty. Yellow bars represent the initial rotational error, while green bars are the final error. The rotational error is computed in quaternions.

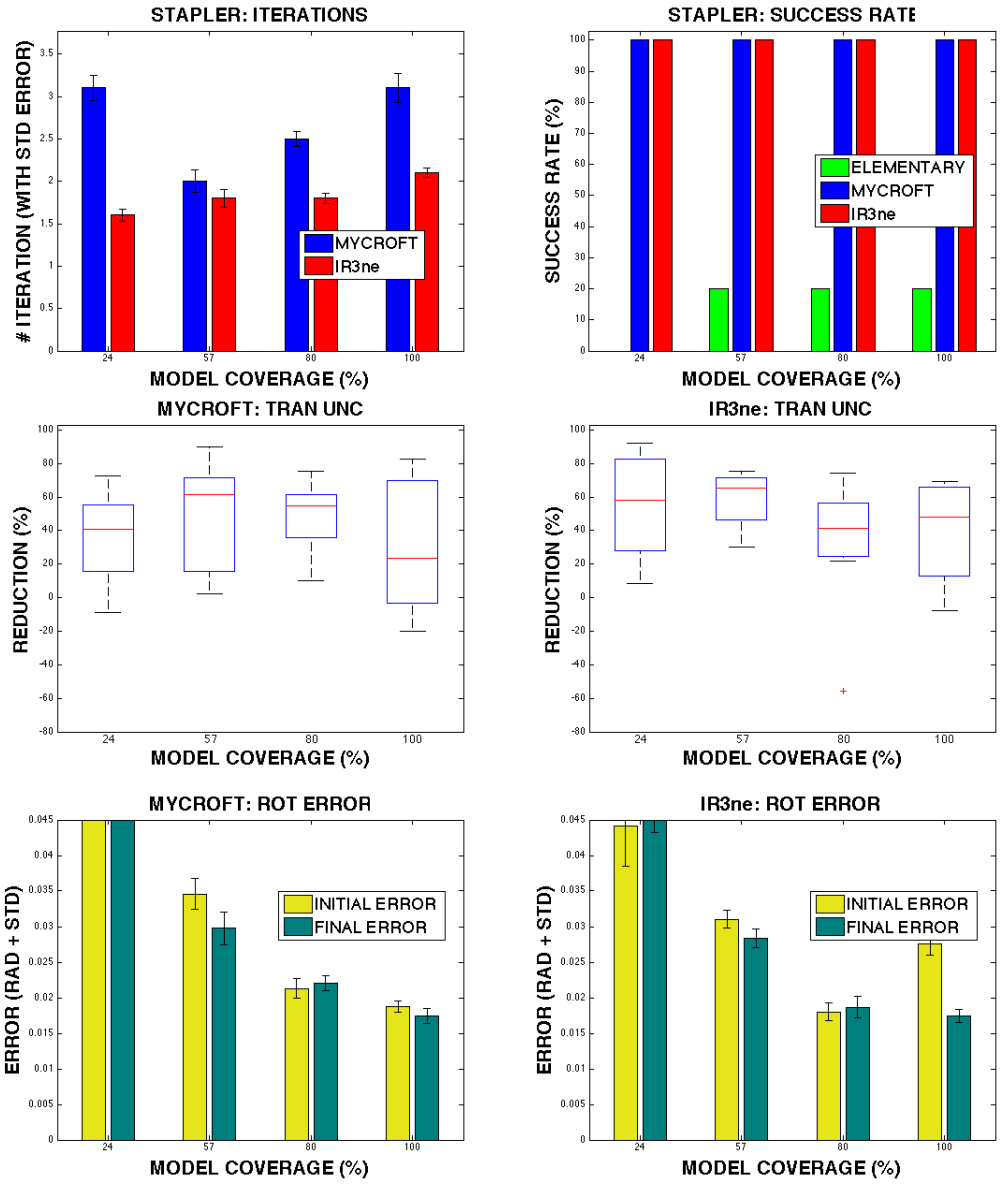


Figure 7.11: Simulated results for 120 trials on a stapler. Three strategies were used: ELEMENTARY (green), MYCROFT (blue), IR3ne (red). All the results are plotted against the initial percentage of model coverage for a total of four conditions. The top left chart presents the average number of iterations across the trials to reach a grasp. The top right chart shows the success rates across the trials. The middle row shows the linear uncertainty reduction (in percentile), while the bottom row presents the reduction in the rotational uncertainty. Yellow bars represent the initial rotational error, while green bars are the final error. The rotational error is computed in quaternions.

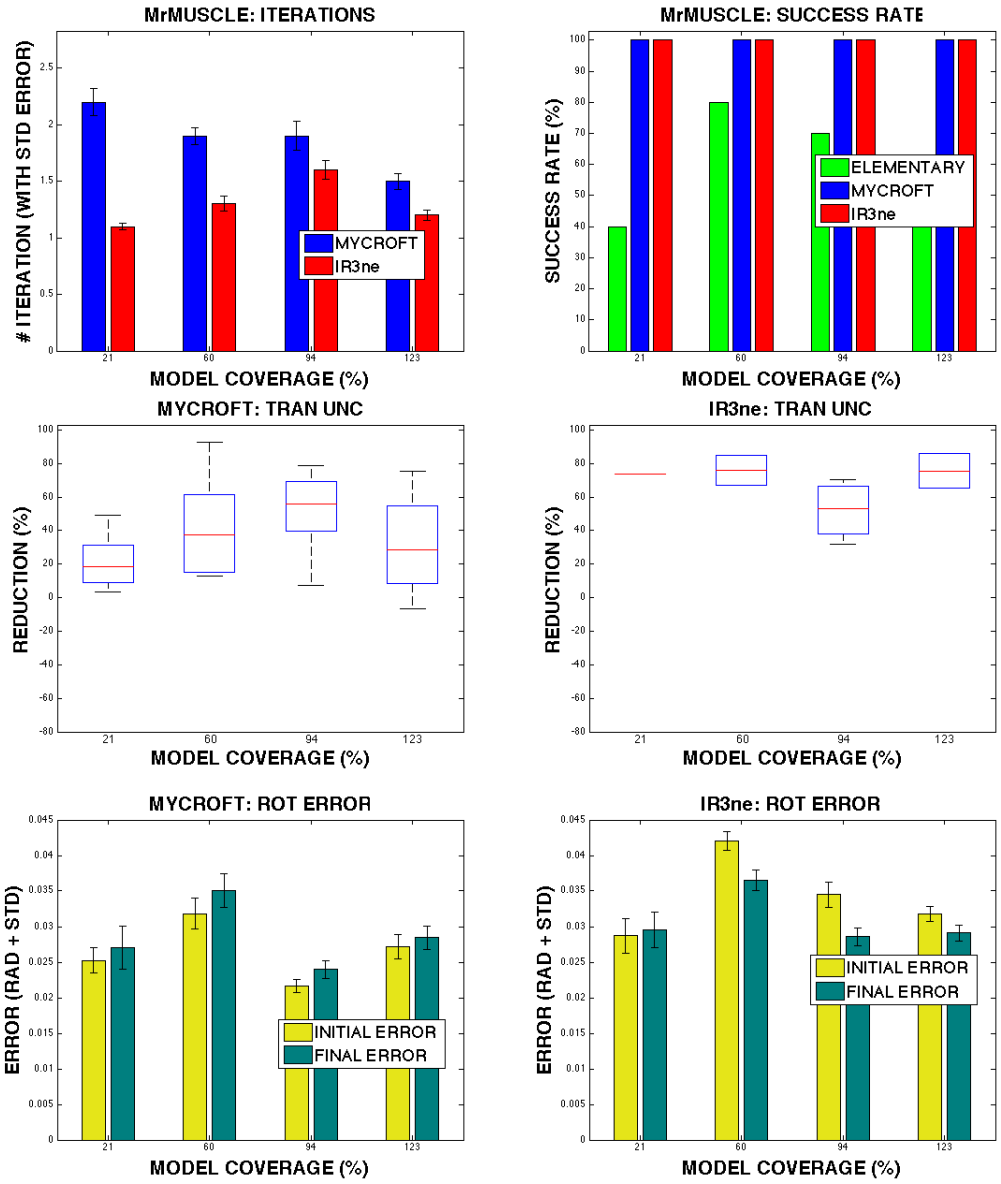


Figure 7.12: Simulated results for 120 trials on a mr muscle spray. Three strategies were used: ELEMENTARY (green), MYCROFT (blue), IR3ne (red). All the results are plotted against the initial percentage of model coverage for a total of four conditions. The top left chart presents the average number of iterations across the trials to reach a grasp. The top right chart shows the success rates across the trials. The middle row shows the linear uncertainty reduction (in percentile), while the bottom row presents the reduction in the rotational uncertainty. Yellow bars represent the initial rotational error, while green bars are the final error. The rotational error is computed in quaternions.

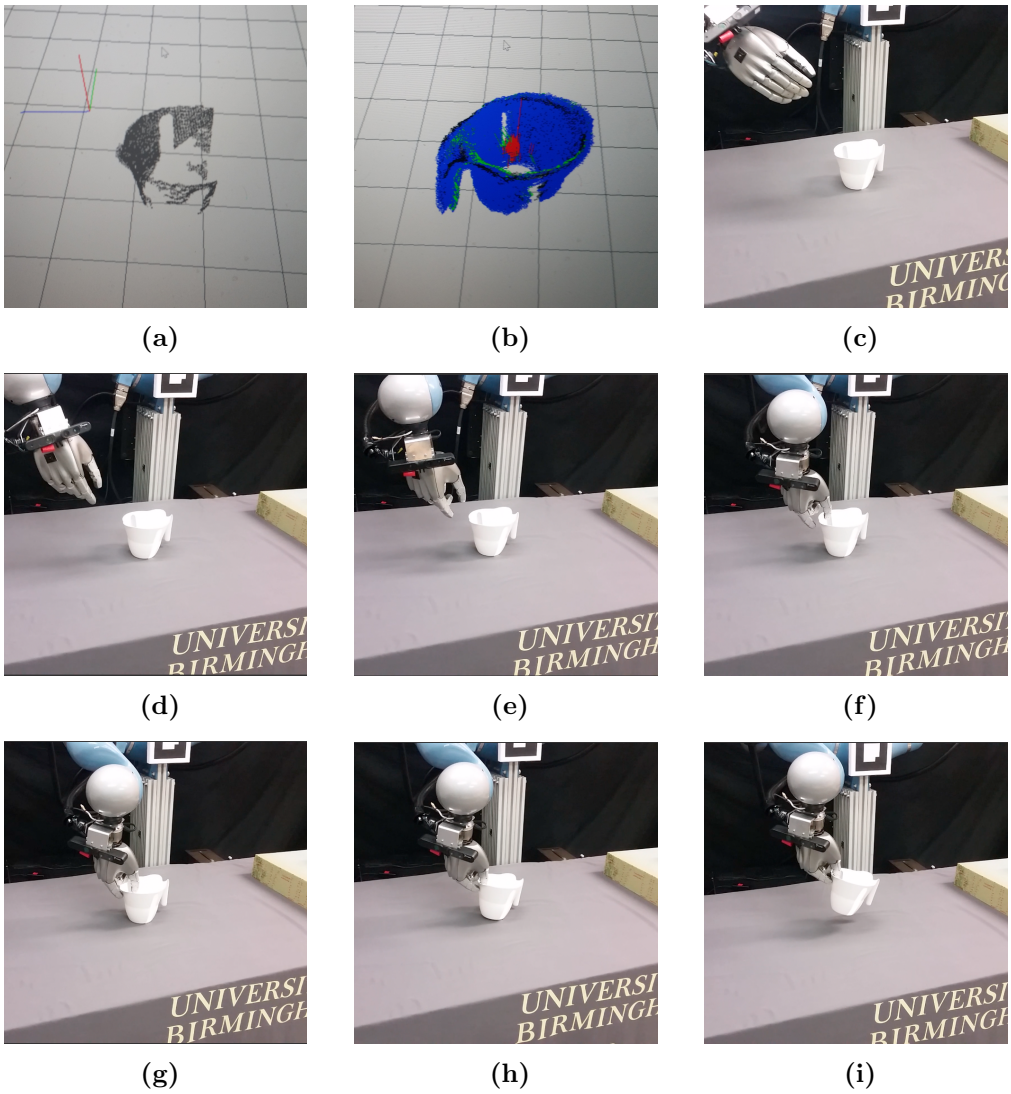


Figure 7.13: Example of plan execution for IR3ne. The top row shows: (a) the partial query point cloud (3 merged views), (b) the belief state (as sub-sampled hypotheses (blue), mean pose (green) and ground truth (black)), (c) the real pose of the object. This example shows the worst case in which the the query point cloud covers only a 22.5% of the model and the handle is not visible. Note that the ground truth (b) (black point cloud) is also estimated with the wrong orientation. Nevertheless, IR3ne executes the planned trajectory (middle row) and achieve e grasp (g) and (h). In (i), Boris successfully lifts the jug.

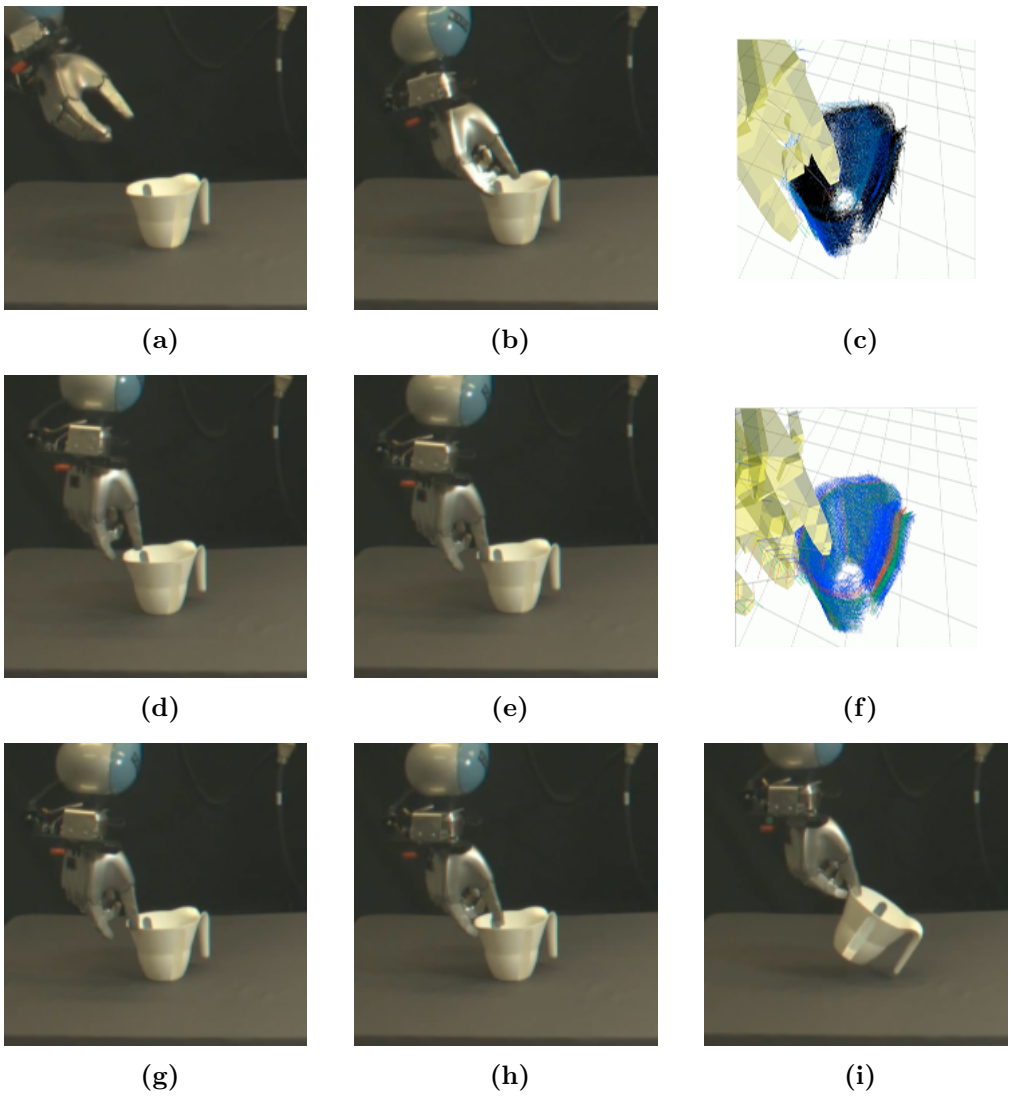


Figure 7.14: The sequence of images presents an interesting case in which the initial belief state does not cover the ground truth, however IR3ne is capable of converging to a grasp in 3 attempts. Each row shows a real robot's attempt and either the belief update after each attempt or the final grasp. The simulated images show the belief update as PF. The colour of each hypothesis is associated with its likelihood, from red (high likelihood) to black (zero likelihood).Top row: due to an erroneous localisation the first grasp attempt collides with the plastic jug on the rim (b). The contact is used to refine the belief state but, since none of the hypotheses match the contact, the hypotheses have all low probability associated (c). Middle row: a second attempt fails but this time the contact allows Boris to localise the object and, finally, to grasp it (third row).

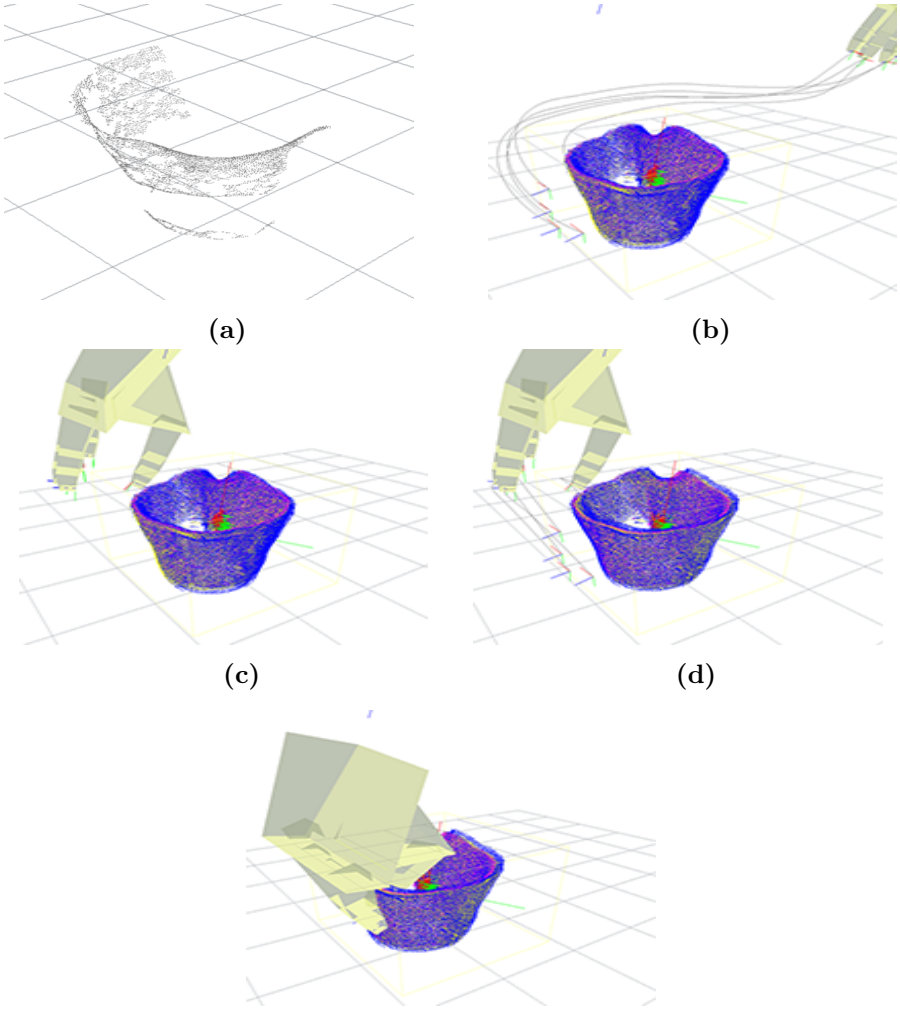


Figure 7.15: Example of plan execution for IR3ne. Top row: a partial point cloud is acquired (a) and a density distribution represented by a set of particles is computed and a reach-to-grasp trajectory is planned (b). The red point cloud identifies the ground truth which matches the partial point cloud in (a) exactly. The blue point clouds represent the low dimensional belief states sub-sampled from the corresponding high dimensional belief states. The yellow point cloud represents the estimated pose for the object. Second row: despite the tiny misplacement between the estimate pose and the ground truth, a contact occurs before the grasp configuration with respect to the mean pose (yellow point cloud) could be reached (c). The belief state is therefore updated from the contact (d) and a new trajectory is planned with respect to the new estimate. Bottom row: the hand reaches the target grasp pose (d).

Chapter 8

Conclusion and discussion

This thesis addressed the problem of deriving planning algorithms for problems of robotic manipulation and grasping. Manipulation has two effects: i) a robot action has a physical effect on the object we wish to manipulate, and ii) a robot action also acquires information about the target object. This thesis treated both situations to enable our robot to plan manipulative actions by accounting for physical and informational effects, and to recover from incorrect assumptions.

This thesis presented a formulation for such problems, called SPAM, and planning systems for solving such problems (SPAM-PLAN) which combine the benefits of motion planning techniques and decision-theoretic frameworks. Two instances of the SPAM problem have been treated in this thesis: i) planning for physical effects and ii) planning for informational effects. Preliminary results presented in this thesis have shown the validity of such an approach by solving non-trivial manipulation problems.

The next sections discuss the main contributions of the thesis and future work.

8.1 Planning for physical effects

The first contribution of this thesis is a novel approach for planning in an action space (e.g. the joint space of a robot manipulator) while accounting for action effects in another space (e.g. the space of object configurations), where both spaces are continuous and it is hard to derive inverse models. This planner has been demonstrated for push operations. The goal is then to plan sequences of pushes to move a 3D object from an initial pose to a desired one. The proposed approach extends a randomised motion planner (here an RRT), which spans the object's configuration space to search for a global solution, while simultaneously, at each tree extension of the RRT, a local planner performs a randomised depth first search procedure in the action space of the robot. The local planner makes use of a physics simulator to predict the object's motion. Errors in these predictions are overcome by re-planning at execution time. This assumes perfect sensing abilities in terms of tracking the object. This tracking determines when the object's motion is not the one expected and triggers the re-planning.

Via empirical experiments in a simulated environment, we have seen that the planned action sequences converge on arbitrarily accurate approximations to the desired goal state. It has also been shown empirically how accuracy and computational expense vary with different parameter choices for the algorithm.

The proposed algorithm has some useful properties:

- It efficiently spans the action space on-the-fly to generate a set of possible push candidates for a given object's pose.
- It uses only a forward model (a physics engine) to compute a push plan.
- It is robust against becoming trapped in local minima.
- Uncertainty associated with prediction can be overcome by re-planning.

8.2 Planning for informational effects

This thesis has argued that, in order to deal with imperfect information, we should derive algorithms which can model and explicitly reason about uncertainty, so that we enable our robot to plan grasps that are likely to succeed, and do so by gathering additional information about the object state so as to recover from incorrect assumptions.

A contribution of this thesis is to present a novel approach for the problem of robot grasping, where a robot reasons about object-pose uncertainty while planning a dexterous reach-to-grasp trajectory. This approach is capable of planning reach-to-grasp trajectories autonomously in unstructured environments, with no knowledge of the object we wish to manipulate, apart from a model point cloud. This is done by reasoning on information effects to refine the localisation of the object to be grasped when the object is not in the expected pose. The main novel outcome is thus to enable tactile information gain planning for dexterous, high degree of freedom (DoF) manipulators. This is achieved by using a combination of information gain planning and a hierarchical PRM.

Several technical contributions have been developed for this approach, including:

- Develop a set of algorithms for sequential re-planning and information gain re-planning.
- Encode non-Gaussian object pose uncertainty in 6D and implicitly capture some of the effects of shape incompleteness in a particle-based belief state concerning object pose (Sections 6.3.2 and 7.3.1).
- Refine the expected object pose through tactile information by using an observation model for contact sensing by a multi-finger hand that palpates a 3D object to be grasped (Sec. 6.4.1).

- Interpret the noisy contact sensors of the robot hand to efficiently stop a reach-to-grasp trajectory if a contact occurs (Sec. 7.3.4).
- Efficiently plan collision-free dexterous reach to grasp trajectories for non-convex objects described as point clouds (Sec. 7.3.5).

The validity of this approach has been demonstrated via empirical results on a real robot. Multiple approaches were also evaluated in a simulated environment. These tests have shown that sequential re-planning achieves a greater success rate than single grasp attempts, and trajectories that maximise information gain require fewer re-planning iterations than conventional planning methods before a grasp is achieved.

The main drawback of this approach derives from the choice of representing the object model as a point cloud. The sets of trials on the robot Boris have shown that this approach is limited by the lack of a predictive model for the object's motion after a contact occurs. The lightest contact may move the object to an unstable configuration, where the object is supported by the link of the robot hand in contact. The current belief filter does not incorporate an FM, assuming that tactile contacts have no physical effects. This may lead to unsuccessful trials. Previous work, e.g. [Kopicki, 2010] showed how predictive motion models for pushed objects could be learned. However, it is non-obvious how to extend such methods to cope with objects represented by arbitrary point clouds.

In addition, the shape incompleteness of the object model may affect collision detection during planning, resulting in planned trajectories that pass through the target object. If this happens, the current implementation interprets the contact as an unexpected observation. Such observation is then used to compute a posteriori belief state over the object pose and to trigger the re-planning. As future work, the aim would be to extend the planner to also reason about the likelihood that the unexpected contact was due to erroneous or incomplete shape information. This would result in a SLAM problem,

i.e. simultaneously localising and object while also mapping its shape, during iterative reach-to-grasp trajectories.

8.3 Future work

There are still several open problems in the area of planning for robot manipulation. As we have seen there are two types of uncertainty that affect manipulation problems: i) prediction uncertainty and ii) state uncertainty. This thesis has treated these two cases separately and with several assumptions. Nevertheless, real problems typically present both kinds of uncertainty. This section suggests some interesting open problems that could be tackled in the next future.

8.3.1 Robust action planning

When the pose of the object to be manipulated is unknown, what is a trajectory that is robust?

In the case of planning for dexterous grasping, the experimental results have shown that trajectories that maximise the likelihood of gathering information (ReGrasp+IG or IR3NE) are more likely to achieve a grasp with less iterations than sequential re-planning (ReGrasp or MYCROFT). Typically one or two iterations are required for the IG planners to reach a target configuration. This empirically suggests that reasoning about the uncertainty leads to more robust reach-to-grasp trajectories with respect to object-pose uncertainty. Similarly, planning for physical effects should benefit from incorporating state uncertainty with respect to the initial pose estimate of the object. In this case, the planner should evaluate or optimise the planned sequence of actions by running the trajectory against many possible starting poses. This should result in a trajectory which maximises the likelihood of achieving the task (e.g. moving an object

towards a desired pose).

Additionally, noisy estimates of physical parameters (such as friction) should also be varied while optimising the planned sequence of actions. This would incorporate uncertainty about the physical parameters, and therefore about future states, and the evaluation/optimisation procedure would select trajectories which are more robust with respect to the unknown model of the environment.

Policies or trajectories?

Rather than optimise a particular trajectory, a different approach would be to find a policy which describes the behaviour of the robot pusher in any possible situation it may encounter. The benefit of computing a policy is to reason explicitly about state and prediction uncertainty, and to be able to select an action to gather information, if necessary. Chapter 4 discussed the problem of finding a policy in continuous domains and its limitations. However, many problems become tractable with such a formulation by computing local policies which concentrate the search effort on a (optimally) reachable space from a particular initial condition.

8.3.2 Dexterous grasping for physical and informational effects

Which grasp configuration is more robust with respect to the state uncertainty?

The proposed approach plans dexterous reach-to-grasp trajectories which explicitly reason about the informational effects of unexpected contacts. In contrast, in the formulation presented in this thesis, the procedure to select the target grasp (Sec. 7.3.2) is unaware of the object-pose uncertainty. Including state uncertainty in the grasp selection procedure should increase the robustness of the sequential re-planning strategies, leading to a more efficient system, as suggested by the work of [Nikandrova et al., 2013].

For which degrees of freedom can a grasp planner operate successfully, without a model

of the physical effects?

As we have mentioned earlier, one of the main drawbacks of the proposed system for grasping is that the belief update is not able to cope with physically unstable configurations of the object resulting from unexpected contacts. Incorporating reasoning about physical effects, however, requires us to model the object we wish to grasp with physical properties, such as friction and mass distribution. A motion model for the object can be obtained by using a physics engine or a learned predictor. Note that in mobile robotics, for example, the localisation problem is typically solved by using a particle filter algorithm which makes use of a motion predictor based on the odometry readings. The motion predictor rotates and translates the set of robot poses, according to the motor commands executed, before updating the belief state. However, in manipulation the resulting object motion depends on the relative pose of the object with respect to the experienced contact. This results in a more costly computation that requires us to simulate the object's motion for each hypothesis given a particular hand pose, and consequently a relative contact.

In addition, incorporating uncertainty in the physical effects would allow us to imagine how a particular contact affects the target object, and consequently the pose estimation after the belief update. Reasoning about such effects would result in planning trajectories that discard reach-to-grasp trajectories that are likely to result in contacts for which the physical effects are uncertain. The challenge here is to efficiently incorporate a model for such effects in the planning procedure that would be computationally affordable.

Bibliography

B. Argall, et al. (2009). 'A survey of robot learning from demonstration'. *Robotics and Autonomous Systems* **57**(5):469–483.

S. Barbera, et al. (eds.) (1999). *Handbook of utility theory*, vol. 1. Springer US.

Y. Bekiroglu, et al. (2011). 'Assessing Grasp Stability Based on Learning and Haptic Data'. *IEEE Trans. on Robotics* **27**(3):616–629.

J. Betts (2001). *Pratical methods for optimal control using nonlinear programming*. Siam.

V. Boor, et al. (1999). 'The gaussian sampling strategy for probabilistic roadmap planners'. In *IEEE Proc. Int. Conf. on Robotic and Automation (ICRA) (ICRA)*.

B. Burns & O. Brock (2007). 'Single-query motion planning with utility-guided random trees'. In *IEEE Proc. Int. Conf. on Robotic and Automation (ICRA) (ICRA*.

J. Butterfass, et al. (2004). 'Design and experiences with DLR Hand II'. *World Automation Congress: Tenth Int. Symposium on Robotics with Applications* **15**:105–110.

J. Butterfass, et al. (1998). 'DLR's multisensory articulated hand. Part I: Hard and software architecture'. In *IEEE Proc. Int. Conf. on Robotic and Automation (ICRA)*.

D. J. Cappelleri, et al. (2006). 'Designing open-loop plans for planar micro-

manipulation'. In *IEEE Proc. Int. Conf. on Robotic and Automation (ICRA)*, pp. 637–642.

M. Catalano, et al. (2014). 'Adaptive Synergies for the Design and Control of the Pisa/IIT SoftHand'. *Int. Journal of Robotics Research* **33**(5):768–782.

P. Cheng, et al. (2007). 'Decidability of motion planning with differential constraints'. In *IEEE Proc. Int. Conf. on Robotic and Automation (ICRA) (ICRA)*.

Y. Cheng (1995). 'Mean Shift, Mode Seeking, and Clustering'. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **17**:790–799.

S. Chitta, et al. (2010). 'Tactile Object Class and Internal State Recognition for Mobile Manipulation'. In *IEEE Proc. Int. Conf. on Robotic and Automation (ICRA)*.

V. Christopoulos & P. Schrater (2009). 'Grasping objects with environmentally induced position uncertainty'. In *PLoS Computational Biology*, vol. 5.

C. Corcoran & R. Platt (2010). 'A measurement model for tracking hand-object state during dexterous manipulation'. In *IEEE Proc. Int. Conf. on Robotic and Automation (ICRA)*.

M. R. Cutkosky (1989). 'On grasp choice, grasp models, and the design of hands for manufacturing tasks'. *IEEE Trans. on Robotics and Automation* **5**(3):269–279.

H. Dang, et al. (2011). 'Blind grasping: Stable robotic grasping using tactile feedback and hand kinematics'. In *IEEE Proc. Int. Conf. on Robotic and Automation (ICRA)*.

R. Deimel & P. Brock (2014). 'A Novel Type of Compliant, Underactuated Robotic Hand for Dexterous Grasping'. In *IEEE Proc. of Robotics: Science and Systems (RSS)*.

R. Detry (2010). *Learning of Multi-Dimensional, Multi-Modal Features for Robotic Grasping*. Ph.D. thesis, University of Liege.

M. Dogar & S. Srinivasa (2010). 'Push-Grasping with Dexterous Hands : Mechanics and a Method'. In *IEEE/RSJ Proc. Int. Conf. on Intelligent Robots and Systems (IROS)*.

A. M. Dollar & R. D. Howe (2008). 'Simple, reliable robotic grasping for human environments'. In *IEEE Int. Conf. on Technologies for Practical Robot Applications (TePRA)*, pp. 156–161.

D. Duff, et al. (2011). 'Physical simulation for monocular 3D model based tracking'. In *IEEE Proc. Int. Conf. on Robotic and Automation (ICRA)*, Shanghai.

S. Ekvall & D. Kragic (2004). 'Interactive grasp learning based on humain demonstration'. In *IEEE Proc. Int. Conf. on Robotics and Automation*.

C. Ferrari & J. Canny (1992). 'Planning optimal grasps'. In *IEEE Proc. Int. Conf. on Robotics and Automation*.

M. Fischer, et al. (1998). 'Learning techniques in a dataglove based telemanipulation system for the DLR Hand'. In *IEEE Proc. Int. Conf. on Robotics and Automation*.

M. Fuchs, et al. (2009). 'Rollin justin - design considerations and realization of a mobile platform for a humanoid upper body'. In *IEEE Proc. Int. Conf. on Robotic and Automation (ICRA)*.

R. Geraerts & M. Overmars (2002). 'A Comparative Study of Probabilistic Roadmap Planners'. In *Workshop on the Algorithmic Foundations of Robotics*.

N. Gorges, et al. (2010). 'Haptic Object Recognition using Passive Joints and Haptic Key Features'. In *IEEE Proc. Int. Conf. on Robotic and Automation (ICRA)*.

G. Grioli, et al. (2012). 'Adaptive synergies: an approach to the design of under-actuated robotic hands'. In *IEEE/RSJ Proc. Int. Conf. on Intelligent Robots and Systems IROS*.

U. Hillenbrand (2008). 'Pose clustering from stereo data'. In *VISAPP Int. Workshop on Robotic Perception*.

U. Hillenbrand & A. Fuchs (2011). 'An experimental study of four variants of pose clustering from dense range data'. In *Computer Vision and Image Understanding*.

K. Hsiao, et al. (2011). 'Bayesian grasp planning'. In *IEEE Proc. Workshop on Mobile Manipulation: Integrating Perception and Manipulation, Int. Conf. on Robotic and Automation (ICRA)*.

K. Hsiao & L. Kaelbling (2010). 'Task-Driven Tactile Exploration'. In *IEEE Proc. of Robotics: Science and Systems (RSS)*.

K. Hsiao, et al. (2007). 'Grasping POMDPs'. In *IEEE Proc. Int. Conf. on Robotic and Automation (ICRA)*.

D. Jacobson & D. Mayne (1970). *Differential dynamic programming*. Elsevier.

M. Jeannerod (1981). 'Intersegmental coordination during reaching at natural visual objects'. In *Attention and Performance IX*.

M. Jeannerod & B. Biguer (1982). 'Visuomotor mechanisms in reaching wiwith extrapersonal space'. In *Advances in the Analysis of Visual Behaviour*, pp. 387–409. MIT.

L. Kaelbling, et al. (1998). 'Planning and Acting in Partially Observable Stochastic Domains'. *Atificial Intelligence* **101**:99–134.

L. Kavraki & P. Svestka (1996). 'Probabilistic roadmaps for path planning in high-dimensional configuration spaces'. In *IEEE Trans. on Robotics and Automation*.

R. L. Klatzky, et al. (1985). 'Identifying objects by touch: An expert system'. *Perception and Psychophysics* **37**:299/302.

L. Kocsis & C. Szepesvari (2006). 'Bandit based Monte-Carlo Planning'. In *15th European Conf. on Machine Learning.*

M. Kopicki (2010). *Prediction learning in robotic manipulation.* Ph.D. thesis, University of Birmingham.

M. Kopicki, et al. (2015). 'One shot learning and generation of dexterous grasps for novel objects'. *The International Journal of Robotics Research* .

M. Kopicki, et al. (2014). 'Learning Dexterous Grasps That Generalise To Novel Objects By Combining Hand And Contact Models'. In *IEEE Proc. Int. Conf. on Robotic and Automation (ICRA).*

M. Kopicki, et al. (2010). 'Predicting workpiece motions under pushing manipulations using the principle of minimum energy'. In *IEEE Proc. of Robotics: Science and Systems (RSS), Workshop.*

M. Kopicki, et al. (2011). 'Learning to predict how rigid objects behave under simple manipulation'. In *IEEE Proc. Int. Conf. on Robotic and Automation (ICRA)*, Shanghai.

K. P. Körding & D. M. Wolpert (2004). 'Probabilistic Inference in Human Sensorimotor Processing'. *Journal of Neurophysiology* **92**:3161–3165.

K. P. Körding & D. M. Wolpert (2006). 'Bayesian decision theory in sensorimotor control'. *Trends Cogn Sci* **10**:319–326.

F. Kyota, et al. (2005). 'Detection and evaluation of grasping positions for autonomous agents'. In *Int. Conf. on Cyberworlds.*

M. S. Landy & D. M. Wolpert (1012). 'Motor control is decision-making'. *Current Opinions in Neurobiology* **22**:1–8.

S. M. LaValle (1998). 'Rapidly-exploring random trees: A new tool for path planning'. Tech. rep., Computer Science Dept, Iowa State University.

S. M. LaValle (2001). 'Rapidly-exploring random trees: Progress and prospects'. In *Algorithmic and computational robotics: new directions: the fourth Workshop on the Algorithmic Foundations of Robotics*, p. 293. AK Peters, Ltd.

S. M. LaValle (2006). *Planning Algorithms*. Cambridge University Press.

S. J. Lederman & R. L. Klatzky (1993). 'Extracting object properties through haptic exploration'. *Acta Psychologica* **84**:29–40.

N. Lepora & K. Gurney (2012). 'The basal ganglia optimize decision making over general perceptual hypotheses'. *Neural Computation* **24**.

N. Lepora, et al. (2013). 'Active Bayesian perception for simultaneous object localization and identification'. In *IEEE Proc. of Robotics: Science and Systems (RSS)*.

M. Littman (1996). *Algorithms for Sequential Decision Marking*. Ph.D. thesis, Department of Computer Science, Brown University.

K. Lynch (1992). 'The Mechanics of Fine Manipulation by Pushing'. In *IEEE Proc. Int. Conf. on Robotic and Automation (ICRA)*, pp. 2269–2276.

O. Madani, et al. (1999). 'On the Undecibility of Probabilistic Planning and Infinite-horizon Partially Observable Markov Decision Problem'. In *16th AAAI-99*.

R. G. Marteniuk, et al. (1990). 'Functspace relationships between grasp and transport components in a prehension Task'. *Hum Move Sci* **9**:149–176.

M. T. Mason (1982). *Manipulator grasping and pushing operations*. PhD thesis, MIT.

M. T. Mason (2001). *Mechanics of robotic manipulation*. MIT press.

N. Melchior & R. Simmons (2007). 'Particle RRT for Path Planning with Uncertainty'. In *IEEE Proc. Int. Conf. on Robotic and Automation (ICRA)*.

T. Meriçli, et al. (2014). 'Push-Manipulation of Complex Passive Mobile Objects using Experimentally Acquired Motion Models'. *Autonomous Robots* pp. 1–13.

B. Mishra (1995). 'Grasp metrics: optimality and complexity'. In *First Workshop on Algorithmic Foundations of Robotics, WAFR*.

B. Mitchinson, et al. (2011). 'Active vibrissal sensing in rodents and marsupials'. *Philos Trans R Soc Lond B Biol Sci* **366**:3037–3048.

M. Muja & D. Lowe (2014). 'Scalable Nearest Neighbor Algorithms for High Dimensional Data'. *Pattern Analysis and Machine Intelligence (PAMI)* **36**:2227–2240.

J. R. Napier (1956). 'The prehensile movements of the human hand'. *j Bone Joint Surg* **38B**:902–913.

Neuronics AG (2004). 'Katana User Manual and Technical Description'.

E. Nikandrova, et al. (2013). 'Towards informative sensor-based grasp planning'. *Robotics and Autonomous Systems* pp. 340–354.

D. A. Novak & J. Hermsdörfer (2009). *Sensorimotor Control of Grasping Physiology and Pathophysiology*. Cambridge University Press.

C. Papadimitriou & J. Tsitsiklis (1987). 'The Complexity of Markovian Decision Processes'. In *Mathematics of Operations Research*.

R. Pelossof, et al. (2004). 'An SVM learning approach to robotic grasping'. In *IEEE Proc. Int. Conf. on Robotics and Automation*.

M. A. Peshkin & A. C. Sanderson (1988). 'The motion of a pushed, sliding workpiece'. *IEEE Journal on Robotics and Automation* **4**:569–598.

A. Petrovskaya & O. Khatib (2011). 'Global localization of objects via touch'. *IEEE Trans. on Robotics* **27**(3):569–585.

J. Pineau, et al. (2006). 'Anytime point-based approximations for large POMDPs'. *Journal of Artificial Intelligence Research* **27**:335–380.

R. Platt, et al. (2001). 'Simultanious localization and grasping as a belief space control problem'. Tech. rep., CSAIL, MIT.

R. Platt, et al. (2012). 'Non-Gaussian Belief Space Planning: Correctness and Complexity'. In *IEEE Proc. Int. Conf. on Robotic and Automation (ICRA)*.

S. Prentice & N. Roy (2008). 'The belief roadmap: Efficient planning in linear POMDPs by factoring the covariance'. In *12th Int. Symposium of Robotics Research*.

M. Puterman (1994). *Markov Decision Processes. Discrete Stochastic Dynamic Programming*. John Wiley & son.

J. H. Reif (1979). 'Complexity of the mover's problem and generalizations'. *Foundation of Computer Science* pp. 421–427.

D. Rosen & J. Kuffner (2008). 'OpenRAVE: A Planning Architecture for Autonomous Robotics'. Tech. Rep. CMU-RI-TR-08-34, Robotics Institute, Carnegie Mellon University.

S. Ross, et al. (2008). 'Online Planning Algorithms for POMDPs'. *Artificial Intelligence Research* **32**:663–704.

R. A. Russell (2000). 'Object recognition by a smart tactile sensor'. In *In Proc. of the Australian Conf. on Robotics and Automation*.

A. Sahbani, et al. (2012). 'An overview of 3D object grasp synthesis algorithms'. *Robotics and Autonomous Systems* **60**(3):326–336.

A. Saxena, et al. (2008). 'Robotic grasping of novel objects using vision'. *Int. Journal of Robotics Research* **27**(2):157–173.

ShadowRobotCompany (2003). 'Design of a dextrous hand for advanced CLAWAR applications'. Tech. rep., Shadow Robot Company.

K. B. Shimoga (1996). 'Robot grasp synthesis algorithms: A survey'. *Int. J. Robotics Research* **15**(3):230–266.

B. Siciliano, et al. (2008). *Robotics: Modelling, Planning and Control*. Springer.

D. Silver & J. Veness (2010). 'Monte-Carlo Planning in Large POMDPs'. In *Advances in Neural Information Processing Systems*.

A. Sloman (2006). 'Polyflaps as a domain for perceiving, acting and learning in a 3-D world'. In *Position Papers for 2006 AAAI Fellows Symposium*, Menlo Park, CA.

R. Smallwood & E. Sondik (1973). 'The Optimal Control of Partial Observable Markov Processes over a Finite Horizon'. In *Operation Research*.

M. Spaan & N. Vlassis (2005). 'Perseus: randomized point-based value iteration for POMDPs'. *Journal of Artificial Intelligence Research* **24**:195–220.

R. Suarez, et al. (2006). 'Grasp quality measures'. Tech. Rep. IOC-DT-P-2006-10, Technical University of Catalunya (UPC).

S. Thrun (2000). 'Monte Carlo POMDPs'. In *Advances in Neural Information Processing Systems 12*, pp. 1064–1070. MIT Press.

G. Tonietti, et al. (2005). 'Design and Control of a Variable Stiffness Actuator for Safe and Fast Physical Human/Robot Interaction'. In *IEEE Proc. Int. Conf. on Robotics and Automation*.

M. Toussaint, et al. (2010). 'Integrated motor control, planning, grasping and high-level

reasoning in a blocks world using probabilistic inference'. In *IEEE Proc. Int. Conf. on Robotic and Automation (ICRA)*.

M. Toussaint, et al. (2011). 'Expectation-Maximization methods for solving (PO)MDPs and optimal control problems'. In *Bayesian Time Series Models*. Cambridge University Press.

O. Tuzel, et al. (2005). 'Simultaneous multiple 3D motion estimation via mode finding on Lie groups'. In *Int. Conf. Computer Vision*.

C. Urmson & R. Simmons (2003). 'Approaches for heuristically biasing RRT growth'. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*.

J. Weisz & P. Allen (2012). 'Pose error robust grasping from contact wrench space metrics'. In *IEEE Proc. Int. Conf. on Robotic and Automation (ICRA)*.

S. A. Wilmarth, et al. (1999). 'MAPRM: a probabilistic roadmap planner with sampling on the medialaxis of the free space'. In *IEEE Proc. Int. Conf. on Robotic and Automation (ICRA) (ICRA*.

J. Wyatt (1997). *Exploration and Inference in Learning from Reinforcement*. Ph.D. thesis, University of Edinburgh.

J. Zheng, et al. (2011). 'An investigation of grasp type and frequency in daily household and machine shop tasks'. In *IEEE Proc. Int. Conf. on Robotic and Automation (ICRA)*.

C. Zito, et al. (2013a). 'Sequential re-planning for dextrous grasping under object-pose uncertainty'. In *Proc. Workshop on Manipulation with Uncertain Models, Robotics: Science and Systems (RSS)*.

C. Zito, et al. (2013b). 'Sequential trajectory re-planning with tactile information gain

for dextrous grasping under object-pose uncertainty'. In *IEEE Proc. Intelligent Robots and Systems (IROS)*.

C. Zito, et al. (2012a). 'Exploratory reach-to-grasp trajectories for uncertain object poses.'. In *Proc. Workshop on Beyond Robot Grasping: Modern Approaches for Dynamic Manipulation. Intelligent Robots and Systems (IROS)*.

C. Zito, et al. (2012b). 'Two-level RRT Planner for Robotic Push Manipulation'. In *IEEE Proc. Intelligent Robots and Systems (IROS)*.